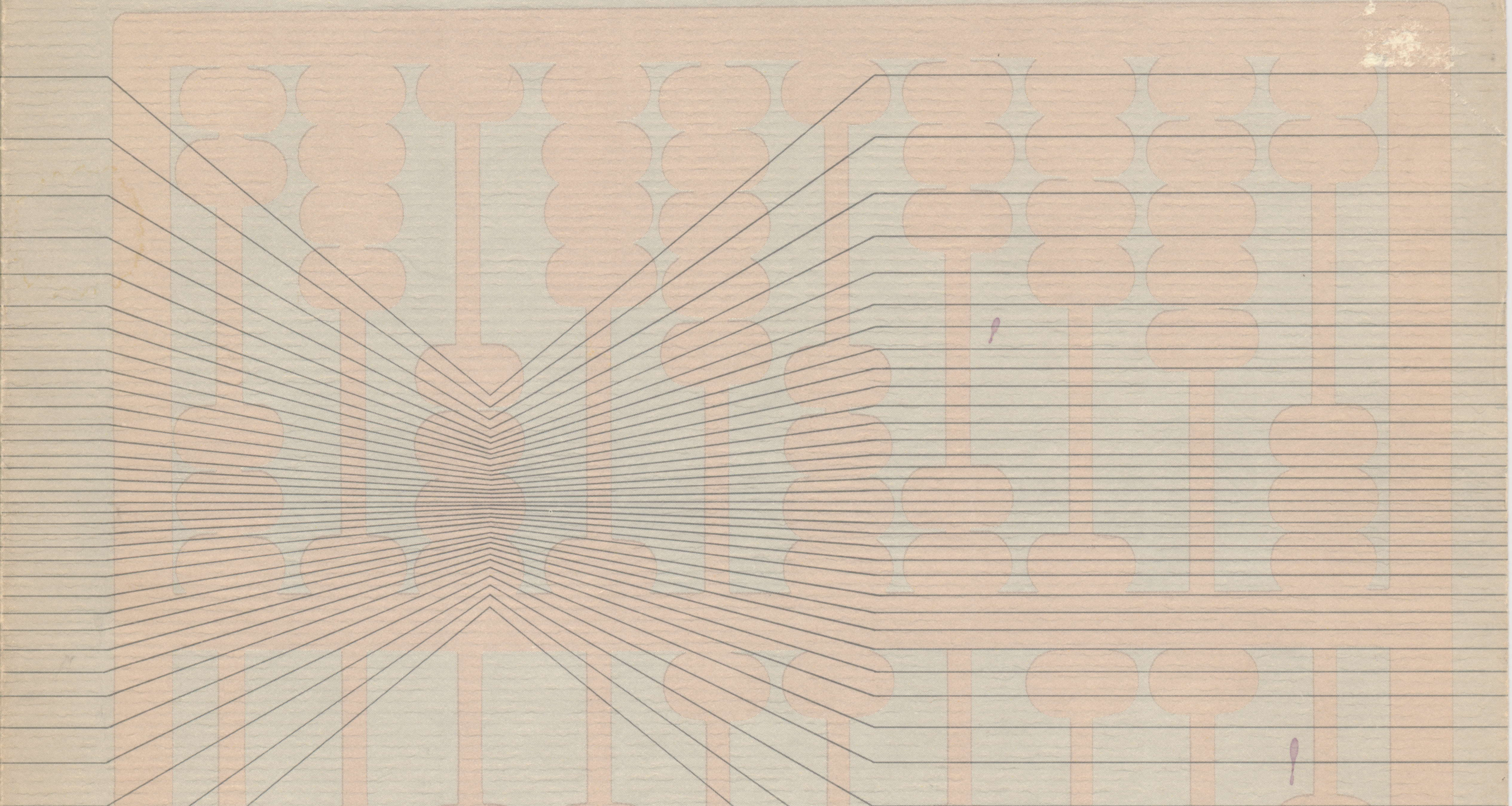
I.P. Sharp Associates

MABRA



MABRA

A RECORD ADMINISTRATION PROGRAM

Copyright March 1980.

All rights reserved. This book or parts thereof may not be reproduced by any means without the written permission of I.P. Sharp Associates Limited.

Printed in Canada.
Publication Code: 80MA03

CONTENTS

Page

Introduction to MABRA 1

System Mode 4
- Creating a MABRA System - *CREATE* 5
- Listing MABRA Systems - *LIST* 5
- Renaming a MABRA System - *RENAME* 6
- Erasing a MABRA System - *ERASE* 6
- Entering Data Mode - *ENTER* 7

Data Mode 8
- Field Management Commands 8
  - Adding Fields - *AF* 8
  - Listing Fields - *LF* 10
  - Deleting Fields - *DF* 11
  - Changing Field Definitions - *CF* 11
- Record Management Commands 14
  - Adding Records - *AR* 14
  - MABRA Constraints 16
  - Listing Records - *LR* 20
  - Changing Records That Have the Same Values in a Field - *CR* 21
  - Changing Records That Have Different Values in a Field - *CD* 23
  - Deleting Records - *DR* 24
  - Batch Printing of Records 25
  - Initiating an Update Task - *UP* 27
  - Taking a New Snapshot - *NS* 28
  - Withdrawing an Update Request - *WU* 29
  - Listing an Update Request - *LU* 29

Crosstabulation Analysis 31
- Crosstabulation Parameters 32
- Running a Crosstabulation Analysis - *AN* 32

Coded Fields 35
- Changing the Code Definition of a Field - *CF* 35
- Adding a Code to the Code-Description Table - *AC* 37
- Listing a Code-Description Table - *LC* 38
- Deleting a Code from the Code-Description Table - *DC* 38

Miscellaneous Commands 40
- Reblocking the Master File - *RB* 40
- Resizing the Master File - *RS* 41
- Copying a MABRA System - *CS* 42
- Field Ordering for *AR* - *FO* 43
- Displaying the Status of a MABRA System - *ST* 45

Multi-User MABRA 47
- Setting Access - *SA* 48
- Listing Access - *LA* 50
- Removing Access - *RA* 50
- Changing Access - *CA* 51
- Entering a New Sign-on Password - *NEWPASS* 54

- How to Save Money 55
  - Using Absolute Record Numbers 55
  - Using *PREVIOUS* in a Constraint 56
  - Inverting Fields 56
  - Reblocking the Master File 56
  - Stacking Input 56
  - Using *CR* Versus *CD* 57
  - Organizing Your Updates 57
  - Setting Field Orders 57

- Examples Showing How to Use 86 *MABRA* 58
  - Basic Commands 58
  - Performing Crosstabulations with *AN* 69
  - Coding Fields with *CF* 80
  - How Multi-User MABRA Works 85

- The Utility Workspace—85 *MABRAUTIL* 90
  - Functions 91
    - *ENTER* 91
    - *SEARCHFOR* 93
    - *READ* 93
    - *APPEND* 95
    - *REPLACE* 95
    - *DELETE* 97
    - *EXIT* 98
    - *GETDESC* 98
    - *GRADE* 99
    - *REORD* 99
    - *DDMMYY* 100
  - How to Change a Field's Data Type 101

- MABRA File Documentation 103
  - First Level: the System Directory File - *SYS* 103
  - Second Level: the *DIR*, *MAS*, *INV*, *ACC*, and *UPD* Files 103
    - Miscellaneous Information File - *DIR* 104
    - Master File - *MAS* 106
    - Inverted Field File - *INV* 107
    - Access File - *ACC* 107
    - Update Request File - *UPD* 108
  - Third Level: the Mask File - *MSK* 108

- Related Fields 110
  - Outline of the Update Task 110
  - Component 17 of the *DIR* File 110
  - A Sample Session 112
  - Debugging Hints 114

- Field Input Validation 115
  - Validation of Data by Type 115
  - User-Definable Functions 115
  - Input Validation in a Sample System 116
  - Sample Function Definitions 116

- APL Constraints 118

- Index 119

INTRODUCTION TO MABRA

MABRA is a computerized program that provides a generalized record-administration facility. This means that for anything you can conceptualize as a record of data, you can easily manage a large number of such records with MABRA.

What can you conceptualize as a record of data? Take anything that requires a common treatment within an organized system: an employee within a personnel system, a stock item within a warehouse system, a product within a manufacturing system, a commodity within an exchange system, a property within a real estate system. Each of these items is the unit of reference within an organized system.

In each of these systems, the units, or records, have attributes in common: each employee in a personnel system has a name, an age, a salary, and a department; each product in a manufacturing system has a model number, an assembly line number, a quality grade, and a retail destination; each property within a real estate system has an address, a price, a market classification, and a sales agent.

MABRA, therefore, can manage a collection of individual records which have the same attributes. When organizing the given attributes of a record, MABRA designates a box—called a field—for each attribute. Within each field, you assign a value, according to the attribute of the field.

To get an idea of how MABRA structures data, picture a grid, or matrix, that has two coordinates: vertical and horizontal. The vertical coordinate contains rows of data which represent **records**. The horizontal coordinate contains columns which represent the **fields** of a record. For example, a MABRA system that stores records of personnel data might contain the fields: *NAME*, *AGE*, *SALARY*, and *DEPT*. Each record within this system would correspond to a particular employee and would contain a different value for each of the four fields.

The table below contains four fields (columns), by one record (a row) of data. Each field represents a particular attribute of the record; the record contains specific values for each field.

4 fields

	NAME	*AGE*	*SALARY*	*DEPT*
1 record	*SMITH*	45	30000	*SALES*

Being a computerized program, MABRA involves interaction between different levels of control. The first level is the **APL system's** immediate-execution mode. At this level, you sign on the APL system to reach the MABRA program; you also return to this level when you leave MABRA. The second level of control is the **MABRA program**. Once you're on this level, the MABRA program (or just plain MABRA) controls everything you do, including what you do on the third level: within a **specific MABRA system**.

As a computerized record-administration program, MABRA offers many worthwhile features:

- A single MABRA system accommodates up to 150,000 records, having as many as 200 fields.
- A single user number accommodates a total of 99 different MABRA systems.
- Records can contain fields that have different types of data: a field's type can be character, integer, real, binary, or date.

- You don't have to be a programmer to succeed with MABRA. MABRA is **easy to use** and provides a high level of on-line assistance. When you enter any command, MABRA issues straightforward prompts for any information needed to carry out the task you have requested. You shouldn't worry about responding to MABRA incorrectly and thereby disrupting your session. If MABRA doesn't understand your response, MABRA prompts for a new response after explaining what is unacceptable with your original response.

  If at any time you are unsure about how to respond to any prompt, simply enter *HELP* for on-line assistance. Enter *HELP* a second time for more detailed assistance. After relevant information prints, MABRA returns to its last prompt and awaits your input.

  If you respond to any prompt with *QUIT*, you exit from MABRA to APL's immediate-execution mode.

- When retrieving data from MABRA, you can use any field of a record to act as a **search key**.

- You can store a field's data in **coded**, or abbreviated, form. A facility enables you to make a table that contains both full descriptions and their corresponding codes for each **coded** field. You can save considerable file space by storing the values for a given field in coded form. At the same time, you can opt for full descriptions when you wish to print reports.

- MABRA systems can be **shared** among designated users. The user who creates a given system is called the *CREATOR*. The *CREATOR* can give different types of access to one or more users. By default, MABRA assumes that the *CREATOR* is the only person whose user number can have access to a particular system. However, **if you are given access** to another user's system, you simply must supply MABRA with the person's user number in order to sign on his or her system.

- A **crosstabulation** feature enables you to perform various analyses on specified groups of records.

- A **utility workspace**, 85 *MABRAUTIL*, exists for the programmer who needs to manipulate raw data within a MABRA system. 85 *MABRAUTIL* is useful, for example, when a MABRA system requires special reports or update functions.

- You can **stack** your field input by entering multiple commands and responses on the same line. Stacking your input saves time and effort when you are working with a large number of records. You must use the diamond (◇) feature to separate each command or response when stacking input. For example, the normal way of replying to the field prompts issued for the *AR* (add record) command is:

  ```
  NAME      : SMITH
  INIT      : CC
  SEX       : F
  DOBIRTH   :
  ```

  Here's how you would stack your input:

  ```
  NAME      : SMITH◇CC◇F
  DOBIRTH   :
  ```

  (For more information about stacked input, see the chapter, "Saving Money".)

Within the MABRA program, there are two operational states: System Mode and Data Mode.

- System Mode is the state within which you create a MABRA system (among other general functions).
- Data Mode is the state within which you first define the fields of a particular system, and then, enter and manipulate the system's records of data.

  Data Mode actually contains two different sets of commands:

  - Field Management Commands, for defining the fields in a MABRA system.
  - Record Management Commands, for manipulating the records in a MABRA system.

SYSTEM MODE

The MABRA program resides in the public library workspace, `86 MABRA`. To access this workspace, enter the following command:

```
        )LOAD 86 MABRA
```

Next, enter:

```
        START
```

This command puts you into System Mode where you can create a new system or access any system which exists on your user number.

However, if you wish to work with an existing system that belongs to another user number, **before** you type *START*, you must identify that system's *CREATOR* by typing,

```
        CREATOR←user-number
```

where **user-number** is the user number of the person whose system you wish to work with. Assuming this user has given you access privileges (see the chapter, "Multi-User MABRA"), you can now work with any system on the specified user number.

MABRA responds to *START* with:

```
* MABRA. VERSION 2
* SYSTEM MODE
:
```

to indicate that you are now in System Mode. The colon (:) is the **command prompt** within System Mode. There are five commands which you can enter in response to this prompt. These are:

CREATE	- create a system
LIST	- list all systems
RENAME	- rename a system
ERASE	- erase a system
ENTER	- access a system

Each of these commands is described in detail below.

To leave MABRA's System Mode and return to the APL system's immediate-execution mode, respond to the command prompt by entering *STOP* or *QUIT*.

1. Creating a MABRA System - *CREATE*

When you enter the *CREATE* command, MABRA creates and initializes the files required for a new MABRA system. To format necessary information about the new system, MABRA first issues the prompts shown below.

```
* ENTER SYSTEM NUMBER (<100 AND >0):
```

You should respond to this prompt by entering an appropriate number for the new system (or by entering *STOP* to return to the command prompt in System Mode). This new system number must be a unique integer (whole number) between 1 and 99. After you enter a valid system number, MABRA prompts with,

```
* ENTER NAME FOR SYSTEM nn:
```

where **nn** is the system number you specified in response to the *ENTER SYSTEM NUMBER* prompt. Enter an appropriate name (up to 40 characters) for the new system (or enter *STOP* to return to the command prompt in System Mode). When you specify a name, MABRA returns with:

```
* ENTER EXPECTED MAXIMUM NUMBER OF RECORDS:
```

Enter an integer between 1 and 150,000 (or enter *STOP* to return to the *ENTER SYSTEM NUMBER* prompt). If you enter a number, it should be a reasonable estimate of the number of records you expect to have in the MABRA system. Note that you can change this number at any time by using the *RS* (resize) command in Data Mode. (See *RS* in the chapter, "Miscellaneous Commands".) When you enter an appropriate number, MABRA responds with,

```
* SYSTEM nn: system-name
* CREATE? (Y/N):
```

where **nn** and **system-name** are the system number and system name you specified above. Enter *Y* to create the new system; enter *N* to prevent its creation and to return to the *ENTER SYSTEM NUMBER* prompt; enter *STOP* to return to the command prompt in System Mode. If you enter *Y*, MABRA prints the message,

```
* SYSTEM CREATED.
```

and then returns you to the command prompt in System Mode.

2. Listing MABRA Systems - *LIST*

You can use this command to list the system numbers and the corresponding names of all MABRA systems owned by the *CREATOR*. When you enter *LIST*, MABRA issues the following prompts:

```
* LIST SYSTEM(S)
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

Here, you should set up the paper at your terminal for output, enter a blank space and press carriage return to start the listing (or enter *STOP* to return to the command prompt in System Mode). Once the listing finishes printing, MABRA waits for you to enter another space and carriage return. This feature is common to several MABRA commands: it allows you to reposition your paper before continuing your work. After you enter this prompt, MABRA returns to the command prompt in System Mode.

3. Renaming a MABRA System - *RENAME*

This command changes the name of an existing MABRA system. If you enter *RENAME*, MABRA prompts with:

```
* RENAME SYSTEM
* SYSTEM NUMBER TO BE RENAMED:
```

Enter the number of the MABRA system whose name you wish to change (or enter *STOP* to return to the command prompt in System Mode). When you specify a system number, MABRA prompts with:

```
* NEW NAME:
```

You should enter the new name for the MABRA system (or enter *STOP* to return to the *SYSTEM NUMBER TO BE RENAMED* prompt). When you specify a new name, MABRA responds with,

** YOU WISH TO CHANGE THE NAME OF SYSTEM* **nn** *FROM* **name1** *TO* **name2**.
** OK TO RENAME? (Y/N):*

where **nn** is the system number you specified above, **name1** is the current system name, and **name2** is the new system name you specified above. Enter *Y* to proceed; enter *N* to return to the *SYSTEM NUMBER TO BE RENAMED* prompt; enter *STOP* to cancel the renaming and to return to the command prompt in System Mode. If you enter *Y*, the system issues the message,

```
* DONE.
```

and returns to the command prompt in System Mode.

4. Erasing a MABRA System - *ERASE*

This command deletes an existing MABRA system by erasing its files. When you enter *ERASE*, MABRA prompts with:

```
* ERASE SYSTEM
* ENTER SYSTEM NUMBER OF SYSTEM TO BE ERASED:
```

You should enter the number of the system you wish to erase. When you specify the system number, MABRA responds with,

** THE SYSTEM YOU WISH TO ERASE IS:*
* **nn: system-name**
** ERASE? (Y/N):*

where **nn** is the system number you specified above, and **system-name** is the name of that system. Enter *Y* to confirm that this is the system you want to erase; enter *N* to prevent erasing the system and to return to the *THE SYSTEM YOU WISH TO ERASE IS* prompt; enter *STOP* to return to the command prompt in System Mode. If you enter *Y*, MABRA prompts with:

```
* ARE YOU SURE? (Y/N):
```

This prompt encourages you to consider whether the system number you just entered is the correct one. If the number is correct, enter *Y*—this time to erase the system. Enter *N* to prevent erasing the system and to return to the *THE SYSTEM YOU WISH TO ERASE IS* prompt; enter *STOP* to return to the command prompt in System Mode. If you enter *Y*, MABRA erases the specified system, prints the message,

** SYSTEM ERASED.*

and returns to the command prompt in System Mode.

5. Entering Data Mode - *ENTER*

Whereas System Mode is the state in which a system is **created** (files are created and initialized), Data Mode is the state in which the content of the system is **developed** and **manipulated** (fields and records are added, deleted, changed, listed). You get into Data Mode by executing the *ENTER* command within System Mode. MABRA responds with:

** SYSTEM NUMBER:*

You should enter the system number of the MABRA system you wish to work with (or enter *STOP* to return to the command prompt in System Mode). When you specify a valid number, MABRA issues the message,

** DATA MODE, SYSTEM IS* **nn**: **system-name**

where **nn** is the system number you just specified and **system name** is its corresponding name. MABRA then issues the command prompt in Data Mode and waits for you to enter some command:

nn:

nn represents the number of the system you are currently working with. Once you are in Data Mode, MABRA prompts for all commands in this manner—that is, by printing the number of the system you are currently working with, followed by a colon.

To return to System Mode from Data Mode, enter *STOP* in response to this "**nn:**" prompt.

DATA MODE

After you use the *CREATE* command to create the files required for a new MABRA system, you are ready to define the structure (i.e., establish the required fields) for the new system, and subsequently, to enter the actual records of data. In order to do this, you must first get into Data Mode, by using the *ENTER* command within System Mode. Recall from above that once you are in Data Mode, MABRA prompts for each command with,

nn:

where **nn** is the number of the MABRA system in which you are currently working.

If at any time you wish to return to System Mode, enter *STOP* in response to the command prompt in Data Mode.

There are two sets of commands that establish and manipulate a MABRA system: **field management commands** and **record management commands**.

FIELD MANAGEMENT COMMANDS

Field management commands define and maintain the fields for the current system. There are six attributes that define a field. These **field definitions** are a field's:

- name
- type (character, integer, real, binary, and date)
- width (fixed-length number of elements)
- inversion status (for conserving file space—see "Miscellaneous Commands")
- code status (for enhancing record management—see "Coded Fields")
- output format (left argument to ⎕*FMT* command)

The Field Management Commands are:

AF - add a field to the system
LF - list all fields' definitions
DF - delete a field from the system
CF - change an attribute of a field's definition

1. Adding Fields - *AF*

This command adds new fields to the current system. When you enter *AF*, MABRA responds with:

```
* ADD FIELD
* ENTER FIELD NAME:
```

Enter the name of the field to be added (or enter *STOP* to return to the "**nn**:" command prompt in Data Mode). The field name must consist of 14 alphanumeric characters or less, and its first character must be alphabetic. The field name **cannot** be *ALL*, *HELP*, *PREVIOUS*, *QUIT*, or *STOP*. When you specify a field name, MABRA prompts with:

```
* TYPE? (CHAR/INTE/REAL/BINA/DATE):
```

Enter the type of the new field (or enter *STOP* to return to the *ENTER FIELD NAME* prompt). Valid types are:

CHAR - character (alphanumeric information)
INTE - integer (whole numbers ranging from `-2*31` through `¯1+2*31`)
REAL - real (includes integers and decimal numbers)
BINA - binary (Boolean numbers, being 0 and 1 only)
DATE - dates (in the form DDMMYY, where DD is the day, MM the month, and YY the year) Note that it is possible to enter and retrieve dates in the MMDDYY format. To do this, you must set the *DATEFLAG* variable to the value 1 before entering the MABRA program with *START*. *DATEFLAG*'s values are: 0 for the default DDMMYY format; 1 for the MMDDYY format.

After you specify a valid type, MABRA prompts with:

```
* NUMBER OF ELEMENTS IN FIELD:
```

Enter the width (APL ⍴) of the new field (or enter *STOP* to return to the *ENTER FIELD NAME* prompt). The number of elements comprising the width is defined according to the field type:

CHAR - width is equal to the number of characters (elements for "42 *ELM ST*." equal 10).
INTE - width is equal to the number of integers, **not** digits (elements for "530000" equal 1).
REAL - width is equal to the number of real numbers, **not** digits (elements for "6.5" equal 1).
BINA - width is equal to the number of binary digits (elements for "1 0 1 1 0" equal 5).

Note that if you specify *DATE* as the type for a given field, the *NUMBER OF ELEMENTS IN FIELD* prompt is not issued because the width of a *DATE* field is always 1.

When you specify the number of elements for the current field, MABRA prompts with:

```
* USE DEFAULT OUTPUT FORMAT? (Y/N):
```

The **output format** refers to the use of SHARP APL's report formatting facility, *⎕FMT*, to control the shape of your printed output. (For further information on *⎕FMT*, see the I.P. Sharp manual, "SHARP APL Report Formatting".) The **default** output formats for each type of field are listed below. Note that the **width** is the repetition factor for *⎕FMT*, and is the same value that you specified in response to the *NUMBER OF ELEMENTS IN FIELD* prompt.

Data Type	**Output Format**
CHAR	**width** *A*1
INTE	**width** *I*10
REAL	**width** *E*10.5
BINA	**width** *I*1
DATE	*G*⍞ZZ/ZZ/Z9⍞

Enter *Y* if the default *⎕FMT* argument is suitable; enter *N* if you wish to specify an alternative *⎕FMT* argument; enter *STOP* if you wish to return to the *ENTER FIELD NAME* prompt. If you enter *Y*, MABRA responds with, "*OK TO ADD? (Y/N):*" (described further below). If you enter *N* in response to the *USE DEFAULT*. . . prompt, MABRA issues the prompt:

** ENTER ⎕FMT ARGUMENT*:

You should enter a valid *⎕FMT* phrase, containing a width that fills the **whole** field (or enter *STOP* to return to the *USE DEFAULT...* prompt).

If you specify a valid format, the system prompts with:

** OK TO ADD? (Y/N)*:

Enter *Y* to add the field whose structure you have just defined; enter *N* or *STOP* to indicate that the previous input should be ignored and that you wish to return to the *ENTER FIELD NAME* prompt. If you enter *Y*, MABRA prints the message,

** FIELD ADDED.*

to indicate that the specially defined field has been added to the system.

MABRA returns to the *ENTER FIELD NAME* prompt. You can now define the structure for the next field and continue in this manner until all required fields are defined for the system.

Note that if you add a field after other records already exist, the new field will be given a value of the appropriate **filler** data. This filler is either **zero** for numeric data types, or **blanks** for character data.

Note further that when referencing field names, you can **abbreviate** them by entering as much of a field's name as is necessary to make it unique. If a system has the fields, *LASTNAME* and *SALARY*, you can reference these with *L* and *S*.

2. Listing Fields - *LF*

This command **lists the field definitions** for all fields in the current system. As mentioned earlier, there are six attributes that define a field:

- field name
- field type (character, integer, real, binary, and date)
- field width (fixed-length number of elements)
- inverted status (see the chapter, "Miscellaneous Commands")
- coded status (width of codes' descriptions; see the chapter, "Coded Fields")
- field output format (left argument to *⎕FMT* command)

When you enter *LF*, MABRA responds with:

** LIST FIELD*
** ALIGN PAPER, TYPE SPACE, CR TO START.*

Set up the paper for output, enter a blank space and press carriage return to produce the field listing (or enter *STOP* to return to the command prompt in Data Mode).

If you choose to list the fields, you will see that your listing contains a column with the heading, "*ACCESS*". This is not, in fact, a field definition, but a convenient indication of the access which you have to each field. If you are the *CREATOR*, you have complete access to add, change, and list all fields. If you are the user of another person's MABRA system, you have limited access to fields, as determined by the *CREATOR*. The chapter, "Multi-User Mabra", explains the various levels of field access that the *CREATOR* may set for other users.

After the list finishes printing, again enter a blank space and press carriage return to get back to the command prompt in Data Mode.

3. Deleting Fields - *DF*

This command deletes fields from the current system. When you enter *DF*, MABRA prompts with:

```
* DELETE FIELD(S)
* FIELDS TO BE DELETED:
```

Enter a list of field names, separated by commas, for those fields you wish to remove (or enter *STOP* to return to the command prompt in Data Mode). When you enter the field names, MABRA prompts with,

** DELETE* **field-names**? *(Y/N)*:

where **field-names** is the list of fields you specified at the *FIELDS TO BE DELETED* prompt. Enter *Y* to confirm that **field-names** are the fields to be deleted; enter *N* or *STOP* to prevent the deletions. If you enter *Y*, MABRA prints the message,

```
* DELETED.
```

to indicate that the specified fields have been removed. MABRA then returns to the *FIELDS TO BE DELETED* prompt.

4. Changing Field Definitions - *CF*

You can use this command to change certain field definitions for any field in the current system. The definitions that you can change for a given field are:

- field name
- field width (number of elements)
- inversion status (inverted or not inverted)
- code status (coded or not coded)
- field output format

Note that *CF* does not change a field's data **type**. Changing a field's type requires special treatment, and is explained in the section, How to Change a Field's Data Type, in the chapter, "The Utility Workspace—85 *MABRAUTIL*".

The definitions of field name, field width, and field output format are explained in this chapter's section, Adding Fields - *AF*, above.

An **inverted** field is a field whose data is stored, for a second time, in a more efficient manner on an auxiliary file that optimizes searching. As a recommendation for those systems which contain more than 200 records, any fields regularly used as search keys should be inverted in order to reduce the computer time required for a search.

A **coded** field contains data in an abbreviated form, as a means of reducing a system's storage requirement. To learn how to change the code status of such a field, see the chapter, "Coded Fields".

When you enter the *CF* command, MABRA issues the prompt:

```
* CHANGE FIELD
* FIELD NAME:
```

Enter the name of the field whose definition you wish to change (or enter *STOP* to return to the command prompt in Data Mode). When you enter a field name, MABRA prompts with:

```
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT?
```

You should indicate what definition you wish to change by choosing one of the above options (or enter *STOP* to return to the *FIELD NAME* prompt).

a. If you specify *NAME* in response to this prompt, MABRA returns with:

   ```
   * ENTER NEW FIELD NAME:
   ```

   Specify the new field name (or enter *STOP* to return to the *CHANGE NAME, WIDTH...* prompt). The field name should be 14 alphanumeric characters or less, and its first character must be alphabetic. If you specify a new name, MABRA prints the message,

   ```
   * OK TO CHANGE NAME FROM name1 TO name2? (Y/N):
   ```

   where **name1** is the current field name you wish to change and **name2** is the new field name. If you enter *Y*, the following message prints,

   ```
   * FIELD NAME CHANGED FROM name1 TO name2.
   * CHANGE FIELD=name2
   ```

   confirming that the field name has been changed. MABRA then returns to the *CHANGE NAME, WIDTH...* prompt.

   If you enter *N* to the *OK TO CHANGE NAME...* prompt, MABRA returns to the *ENTER NEW FIELD NAME* prompt.

b. If you specify that you wish to change the width of the field (enter *WIDTH*), MABRA prompts with:

   ```
   * WIDTH IS CURRENTLY: x
   * ENTER NEW WIDTH:
   ```

   You should specify the new number of elements for the field (or enter *STOP* to return to the *CHANGE NAME, WIDTH...* prompt). If the new width you specify is less than the previous width, all values previously stored in this field are truncated along the field's right boundary. If the new width you specify is greater than the previous width, the values in the field are padded with the appropriate filler data along the right boundary. When you specify a new width, MABRA responds with,

   ```
   * YOU WISH TO CHANGE WIDTH FROM x TO y? (Y/N):
   ```

   where **x** is the old width for the field and **y** is the new width you specified above. Enter *Y* to change the width; enter *N* to return to the *ENTER NEW WIDTH* prompt; enter *STOP* to return to the *CHANGE NAME, WIDTH...* prompt. If you enter *Y*, MABRA prints the message,

   ```
   * WIDTH CHANGED; REMEMBER TO CHANGE ⎕FMT ARGUMENT.
   * CHANGE FIELD=field-name
   ```

Where **field-name** represents the field whose width you just changed. You must change the width factor in the field's *⎕FMT* format as a result of changing the field's width here (see section e., below). The new format requires a valid *⎕FMT* phrase, containing a width that fills the whole field.

MABRA then returns to the *CHANGE NAME, WIDTH...* prompt.

Note that if you attempt to change the width for a field which is **inverted**, an error message prints to indicate that the width cannot be changed until the field is **un-inverted**. This condition is explained in the section directly below.

c. If you specify that the inversion status is to be changed (*INVERT*), and the specified field is **not** inverted, MABRA responds with:

** FIELD IS NOT INVERTED.*
** INVERT? (Y/N):*

Enter *Y* to invert the field; enter *N* or *STOP* to return to the *CHANGE NAME, WIDTH...* prompt. If you enter *Y*, MABRA prints the message,

** INVERTED.*

to confirm that the **field-name**'s data has been stored again, this time in an auxiliary file that optimizes searching. MABRA then returns to the *CHANGE NAME, WIDTH...* prompt.

If you enter *INVERT*, and the field in question is **already** inverted, MABRA responds with:

** FIELD IS ALREADY INVERTED.*
** UN-INVERT? (Y/N):*

Note that if the field is not inverted and is used often as a search field, the cost of a search is more expensive. Enter *Y* to remove data from the auxiliary file; enter *N* or *STOP* to return to the *CHANGE NAME, WIDTH...* prompt. If you enter *Y*, MABRA prints the message,

** UN-INVERTED.*

to confirm that the field's inversion status is changed. MABRA then returns to the *CHANGE NAME, WIDTH...* prompt.

d. To learn how to change the **code** definition of a field, see the chapter, "Coded Fields".

e. If you specify that the output format (*⎕FMT*) of the field is to be changed, MABRA prompts with:

** NEW ⎕FMT ARGUMENT:*

Enter a valid *⎕FMT* phrase, containing a width that fills the **whole** field (or enter *STOP* to return to the *CHANGE NAME, WIDTH...* prompt). If you specify an acceptable new format, MABRA prints the message,

** FORMAT CHANGED.*

to confirm the alteration to the *⎕FMT* argument. MABRA then returns to the *CHANGE NAME, WIDTH...* prompt.

RECORD MANAGEMENT COMMANDS

After you define the fields for a given system, you can maintain and analyze the actual records of data for a system by using the appropriate **record management commands**. These are:

AR - add records to the system
LR - list records in the system
CR - change a field's value to the same, new value
CD - change a field's values to different, new values
DR - delete records from the system
BP - use the batch printing facility to list records
UP - initiate a task that updates additions, changes, or deletions of records
NS - display the newest version (take a new snapshot) of Data Mode
WU - withdraw an update task
LU - list specified update tasks

1. Adding Records - *AR*

This command adds new records to the current system. When you enter the *AR* command, MABRA responds by prompting for the value of the first field in the first record you wish to add. For example, suppose you are working with a system which has three fields called *NAME*, *JOB*, and *SALARY*. When you enter *AR*, MABRA prompts for a value for the first field, as shown below:

```
* ADD RECORD(S)
NAME              :
```

MABRA continues to prompt for a value for each field until you build a complete record. When you specify a value for the last field of the first record, MABRA prompts for a value for the first field in the next record you wish to add. MABRA continues in this manner until you enter *STOP* to return to the command prompt in Data Mode.

The *FO* (field ordering) command enhances the *AR* command by letting you specify alternative sequences in which field prompting may occur. For more information, see the *FO* command in the chapter, "Miscellaneous Commands".

You can respond to a prompt for field data in any of the four ways listed below.

a. You can enter the actual value to be stored in the field. If the number of elements in the value you specify is less than the number of elements required for the field, the value is padded with the appropriate filler data along the field's right boundary. If the number of elements in the new value is greater than the number of elements in the original field, the new value is truncated along the right, and a message prints to indicate this truncation. The value is also checked for the validity of its data type:

   Integer - all elements must be valid integers (whole numbers) from -2*31 through ¯1+2*31
   Real - all elements must be valid numerics (integers and decimals)
   Binary - all elements must be 0 or 1 and separated by a blank space
   Date - must be in the form: DDMMYY, MMYY, YY, or 0 where DD≤31, MM≤12, YY≤99, and 0 indicates there is no date

   MABRA does not validate **character** data.

b. If you have no value to enter for a particular field, or if you notice an input error in a previous field (within the current record), you can use the APL branch symbol (→) to go to a desired field. This is MABRA's "jump to" feature. If you enter "**→field-name**" (where **field-name** is the name of some field defined for the current system), MABRA re-prompts for a value for **field-name**.

For example, consider a system with the field names: *NAME*, *SEX*, *DOBIRTH*, *JOB*, and *SALARY*. When entering records of data for this system, you could use the "jump to" feature as shown below:

```
NAME        :SMTIH                          (note the typing mistake in SMITH)
SEX         :M
DOBIRTH     :→JOB
JOB         :CONSULTANT
SALARY      :→NAME
NAME        :SMITH
SEX         :→SALARY
SALARY      :10000

NAME        :
```

First you enter values for the *NAME* and *SEX* fields. Since you have no value to enter for *DOBIRTH*, you choose to jump to the *JOB* field. After you enter a value for this field, you are prompted for a value for *SALARY*. At this point, you notice that you have entered the value for the *NAME* field incorrectly and so you jump back to this field by entering *→NAME*. After you enter the new value, MABRA prompts for a value for the next field (in this case *SEX*). Since you have already entered a value for this field, you jump to the *SALARY* field and enter its value. Now you have completed the input for this record. Consequently, MABRA prompts for a value for the first field (in this case, *NAME*) of the next record.

Note that any unassigned field is given a value of the filler data which is appropriate for that type of field.

c. You can enter a naked branch (→). Such a right-pointing arrow, without a field name following it, signals the end of input for the current record. Any unassigned field is given a value of the appropriate filler for that type of field. After you signal an end to field input, MABRA responds by submitting an update request:

```
* UPDATE REQUEST NO. nn FILED.
```

where **nn** represents the identification number of the update request. MABRA then returns to the *CONSTRAINTS* prompt.

d. You can enter *STOP* to ignore the input for the **current record** and to file an update request for all **previous records** you have entered in the current session. MABRA then returns to the command prompt in Data Mode.

Note that additions are not made to the system until you enter the *UP* (initiate update task) command. (See the section, Initiating an Update Task - *UP*, in this chapter).

2. MABRA Constraints

After you enter records into the system, you may wish to retrieve them in order to display, delete, or make changes to them (using the *LR*, *BP*, *AN*, *DR*, *CR*, and *CD* commands). To retrieve records, you must first define them by specifying a **constraint**. A constraint is **not** an APL expression, but rather, a simple way to identify records. A constraint has the following format:

field-name relation value

field-name represents the name of the field that undergoes comparison. This name can be the first character of the field's name, so long as it uniquely identifies the field under comparison.

relation represents one of the following comparisons:

Symbol	**Meaning**
<	less than
≤	less than or equal to
=	equal to
≠	not equal to
≥	greater than or equal to
>	greater than
⊃	contains

value represents the data to be compared with the content of **field-name**.

To demonstrate how to identify records when using constraints, consider a system having the following fields:

NAME - surname (15 characters)
SEX - sex (1 character)
DOBIRTH - date of birth (date)

The records of data stored for this system are:

NAME	*SEX*	*DOBIRTH*
SMITH	*F*	03/08/12
BLOGGS	*M*	11/03/45
JONES	*M*	01/05/14
BLACK	*M*	23/09/60
MCALPIN	*F*	13/12/55
DOE	*F*	01/02/32
DOYLE	*M*	21/12/57
PIKE	*M*	13/10/58
SMITHSTON	*F*	10/11/42
HYNDMAN	*M*	10/07/55

The following examples demonstrate how constraints can identify records in this system. After you enter a valid constraint, MABRA responds by indicating the number of records in the current system that match the specified constraint.

Note that you cannot use quotations around a value you wish to represent as a character string.

```
    NAME=SMITH
  2 FOUND.
```

The above example locates all records whose *NAME* fields contain values that **start** with the character string *SMITH*. Note that *SMITHSTON* also meets this constraint.

```
    N=SMITHb
  1 FOUND.
```

This constraint locates all records whose *NAME* fields contain values that start with *SMITH***b**, where "**b**" indicates that you enter a blank space by depressing the "space bar" on the terminal. *N* uniquely identifies the field called *NAME*.

```
    SEX=M
  6 FOUND.
```

This example locates all records with *SEX* fields containing values whose first character is equal to *M* (i.e., finds all males).

```
    NAME≥H
  6 FOUND.
```

This locates all records with *NAME* fields containing values whose first letters are greater than or equal to *H* (i.e., finds all people whose names start with the letters *H* through *Z*).

```
    NAME≠JONES
  9 FOUND.
```

This example finds all people whose names do not start with *JONES*.

```
    DOBIRTH=100755
  1 FOUND.
```

This finds all people whose birthdates are July 10, 1955. You must specify the date in the form DDMMYY, MMYY, or YY, where DD represents the day, MM the month, and YY the year.

```
    DOBIRTH=55
  2 FOUND.
```

This constraint finds all people who were born in 1955.

```
    DOBIRTH<0132
  2 FOUND.
```

This example finds all people who were born before January, 1932.

In many cases, you may be interested in identifying records based on an **age** rather than a **date**. For example, if today's date is October 13, 1979, then the constraint,

```
    DOBIRTH<131014
  2 FOUND.
```

finds all people whose birthdates are before October 13, 1914 (or whose **ages** are greater than or equal to 65 years). Rather than calculating the actual date to find all those who are a certain age, you can place an alpha (α) in front of the field name (in this case, *DOBIRTH*) and write a constraint in the manner shown below. Note that the α prefix converts a date to a length of time (in years) from today's date, and can only be applied to fields that are of *DATE* type.

```
    αDOBIRTH≥65
 2 FOUND.
```

This finds all people who are 65 years old or older. Note the difference between the above constraint and the following,

```
    DOBIRTH≥65
 0 FOUND.
```

which finds all people born after 1965.

Sometimes a group of records cannot be identified by a single constraint. In this case, two or more constraints can be specified using the **and** (∧) or the **or** (∨) connectives. **And** indicates that you want to satisfy **both** conditions that border the connective, whereas **or** indicates that you want to satisfy **either** of the two conditions that border the conjunction. For example,

```
    NAME<H∧SEX=M
 3 FOUND.
```

This constraint locates all records whose *NAME* fields contain values for their first letters that are less than *H* (from *A* through *G*), **and** whose sex is male.

```
    NAME≥C∧NAME≤E
 4 FOUND.
```

This finds all people whose names start with the letters *C* through *E*.

```
    SEX=M∨SEX=F
10 FOUND.
```

This example finds all people who are either male **or** female. Predictably, the result contains the total number of records.

```
    NAME=SMITH∨NAME=JONES∧SEX=M
 1 FOUND.
```

This finds all males whose names begin with *SMITH* **or** *JONES*. Note that the expression uses **both connectives** (and, or) and is evaluated from **left** to **right**. That is, after finding the group of people whose names begin with *SMITH* or *JONES*, then all males are selected from this group.

You can use parentheses to group constraints together. Constraints both **within** and **outside of** parentheses may appear in the same expression, but those constraints **within** parentheses are evaluated **first**. For example,

```
    SEX=M∧(NAME=SMITH∨NAME=JONES)
 1 FOUND.
```

This constraint is identical to the previous example. The constraint within the parentheses is evaluated before "anding" with *SEX=M.*

```
     SEX=F∧NAME=SMITH∨NAME=JONES
  3 FOUND.
```

This finds all females whose surnames starts with *SMITH*, or all people whose names start with *JONES*. To clarify this expression you could rewrite it as follows:

```
     (SEX=F∧NAME=SMITH)∨NAME=JONES
  3 FOUND.
```

The constraint,

```
     (SEX=M∧αDOBIRTH>60)∨(SEX=F∧αDOBIRTH>55)
  2 FOUND.
```

finds all people satisfying the manditory U.K. retirement age within the next five years. (U.K. retirement age is 65 for men, and 60 for women.)

MABRA allows you to subscript fields or, in other words, to compare a certain part of a field to a specified value. The following expression, for example,

```
     NAME[2]=O
  3 FOUND.
```

locates all records whose *NAME* fields have a value of "*O*" as their second letters.

```
     NAME[3 4]=KE
  1 FOUND.
```

This example finds all people having *KE* as the third and fourth letters in their names.

```
     SEX[1]=M
  6 FOUND.
```

This is equivalent to the constraint, *SEX=M*. Likewise, *NAME[1]>M* is equivalent to *NAME>M*.

MABRA also allows you to perform a string search, that is, to see if a field **contains** a specified string of values. The expression shown below,

```
     NAME⊃I
  4 FOUND.
```

searches the *NAME* fields of all records for the value *I* in any location.

The next example locates all records whose *NAME* fields contain the character string *MAN* in any location.

```
     NAME⊃MAN
  1 FOUND.
```

Another useful extension to a MABRA constraint is the ability to complement (negate) constraints. The constraint, *SEX=F∧αDOBIRTH>55*, finds all females older than 55 years. To find all those who do not meet this constraint, you can place a "~" (symbol for **not**) in front of this expression. For example,

```
       ~(SEX=F∧αDOBIRTH>55)
   9 FOUND
```

finds all people who are not females older than 55 years.

```
       ~(NAME=SMITH∨NAME=JONES)
   7 FOUND.
```

This finds all people whose names do not begin with *SMITH* or *JONES*.

Note that if you wish to retrieve **all** records in a system, specify *ALL* when you enter a constraint. If you wish to retrieve records based only on those records in the **previous** constraint, specify *PREVIOUS* when you enter the next constraint. For example, the constraint:

```
       PREVIOUS∧SEX=F
   2 FOUND.
```

examines the 7 records found in the ~(*NAME=SMITH∨NAME=JONES*) constraint and finds 2 records whose *SEX* fields are female.

3. Listing Records - *LR*

This command lists records at your terminal after you first identify them with a constraint. You have the option of specifying which fields within these records you wish to print and/or sort. When you enter the *LR* command, MABRA responds by issuing the following prompt for constraints:

```
* LIST RECORD(S)
* CONSTRAINTS:
```

Enter a valid MABRA constraint to identify the records you wish to list; enter *ALL* to list all records; enter *STOP* to return to the command prompt in Data Mode. If you specify a valid constraint, MABRA responds with,

n *FOUND*.

indicating that **n** records have been located which meet the constraint you specified. (If zero records are found, MABRA re-issues the *CONSTRAINTS* prompt.) MABRA then prompts with:

```
* PRINT FIELDS:
```

You should specify which fields, within the located records, you wish to print (or enter *STOP* to return to the *CONSTRAINTS* prompt). The order in which you enter the field names is the order in which they will print. The list can contain **field names or certain field derivatives**, separated by commas, as shown below:

Field Format		Output Description
field-name	-	prints the data from the field represented by **field-name**
αdate-field-name	-	prints the data in **date-field-name** as a conversion into the length of time (in years) from today
(**field-name**)	-	prints the **code** description of the data in **field-name**; for more information, see the chapter, "Coded Fields"
ALL	-	prints the data from all fields in a record
~field-name	-	is normally used in association with *ALL*. For example, if all fields are to be printed except a few, rather than entering all the field names, you can specify: *ALL*,**~field-name**. This results in all fields being printed **except field-name**.

The number of characters required to print the specified fields of a record should not exceed *⎕PW*, having a default value of 132. If more characters occur, a warning prints, saying that the overflow will wrap around to the next line. After you respond appropriately to the *PRINT FIELDS* prompt, MABRA prompts for the fields by which you wish to sort the records:

```
* SORT FIELDS:
```

You can sort the located records by **any** combination of fields (not merely by print fields), with each field in either ascending or descending order. Enter a list of sort fields, separated by commas, if sorting is required; enter a blank space and press carriage return to indicate that no sorting is required; enter *STOP* to return to the *PRINT FIELDS* prompt. Note that the leftmost field in a list of field names is the major sort key, and therefore, the order for **displaying** each record depends on how you choose to sort this leftmost field. For example, if your system has a *SURNAME* field, you can opt to sort it in **ascending** order and thereby display *SURNAME* in alphabetical order. The formats for sorting fields are:

field-name	-	sort values in **field-name** in ascending order
-field-name	-	sort values in **field-name** in descending order

If you requested any print fields, MABRA prompts with the following message to prepare you for output:

```
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

Set up the paper, enter a blank space and press carriage return to produce your list of fields (or enter *STOP* to return the the command prompt in Data Mode). Once the output finishes printing, again enter a blank space and press carriage return to get back to the *CONSTRAINTS* prompt.

4. Changing Records That Have the Same Values in a Field - *CR*

This command changes **all** values in a specified **field** within a **group of records** to the **same, new value**. You must use constraints to locate the records you wish to change. When you enter *CR*, MABRA responds with:

```
* CHANGE RECORDS
* CONSTRAINTS:
```

Enter a valid MABRA constraint to identify the records you wish to modify (or enter *STOP* to return to the command prompt in Data Mode). When you identify the records in question, MABRA responds with,

n *FOUND.*

to indicate that **n** records have been found which match the specified constraint. (If zero records are found, the *CONSTRAINTS* prompt is re-issued.) MABRA then returns with:

```
* PRINT? (Y/N):
```

This prompt gives you the opportunity to look at the located records **before** you change the values in any field. Enter *Y* to print the records. If you do enter *Y*, the system prompts with *PRINT FIELDS* and *SORT FIELDS* in the same manner as with the *LR* command, above. To bypass this process, enter *N*.

MABRA then issues the following prompt:

```
* ENTER CHANGES.
:
```

Enter the first field change to the right of the ":", by using the format,

field-name←value

where **field-name** is the name of the field to be changed, "←" is the APL **assignment** arrow, and **value** is the new value being assigned to that field. Enter one field change per line and type *STOP* after you make all the necessary changes. MABRA then prompts with:

```
* LIST CHANGES? (Y/N):
```

Enter *Y* to list the changes you just made; enter *N* to bypass the listing. MABRA then prompts with:

```
* FILE UPDATE REQUEST? (Y/N):
```

Enter *Y* to file an update **request** for the changes. Note that changes are **not** made to the system until you enter the *UP* command. (See the section, Initiating an Update Task - *UP*, in this chapter.) If you enter *N*, any changes you just made are ignored and MABRA issues the *CONSTRAINTS* prompt again. If you respond with *Y*, MABRA prints the message,

** UPDATE REQUEST NO.* **nn** *FILED.*

indicating that update request number **nn** is pending. MABRA then returns to the *CONSTRAINTS* prompt.

5. Changing Records That Have Different Values in a Field - *CD*

This command changes **individual** values of a specified **field** within **a group of records** to **different, new values**. *CD* differs from *CR* in that *CR* changes **all** values in a specified field within a group of records to one new value, whereas *CD* prompts for the new value of a specified field for **each** record within a group of records. When you enter *CD*, MABRA responds with the following prompt:

```
* CHANGE DIFFERENT VALUES
* CHANGE FIELDS:
```

You should enter the field names, separated by commas, of the fields whose values you wish to change (or enter *STOP* to return to the command prompt in Data Mode). When you enter valid field names, MABRA prompts with:

```
* PRINT FIELDS:
```

Enter the field names, separated by commas, of any fields that will better identify each record before you are prompted for field changes. Take, as an example, a personnel system for which you wish to **change** the *SALARY* and the *JOBTITLE* fields within a group of records. You might first wish to **print** the *SURNAME*, *DOJOIN* (date when the person joined the company), *LOCATION*, and current *SALARY* fields to identify each record before changing it.

After you enter a list of valid print fields, MABRA responds with:

```
* SORT FIELDS:
```

This prompt allows you to order your **print fields** into the sequence that will best help to identify a record just prior to its being changed, for example: *-DOJOIN,SURNAME,LOCATION,SALARY*.

Enter a list of sort fields (according to the format in *LR*, above) if your print fields require sorting; enter a blank space and a carriage return to indicate that no sorting is necessary; enter *STOP* to return to the *CHANGE FIELD* prompt.

If you enter a valid response for sort fields, MABRA then prompts for constraints to identify the records whose fields you wish to change.

```
* CONSTRAINTS:
```

Enter a valid MABRA constraint (or enter *STOP* to return to the *CHANGE FIELDS* prompt). If you specify a valid constraint, MABRA responds with,

n *FOUND.*

to indicate that **n** records exist that meet your constraint. A maximum of 100 records can reside in the workspace at one time. Therefore, if your first constraint locates more than 100 records, the *CONSTRAINTS* prompt is reissued, enabling you to identify a smaller group of records. Recall that you can use the *PREVIOUS* feature to narrow a constraint. After a constraint locates 100 records or less, MABRA responds with:

⋆ **print-field-1;print-field-2;...**
change-field-1:
↓
change-field-n:

This prompt identifies the first record you wish to change. The first line shows the values of the print fields that you entered earlier and, if you entered sort fields, lists the print fields in the order you specified. The subsequent lines are prompts for changes to each field that you specified in the *CHANGE FIELDS* prompt.

When **changing** field values, you must use the same rules for data input as those rules for **adding** field values. See the section, Adding Records - *AR*, in this chapter.

MABRA continues to prompt for a value for each specified field until changes to all such fields are complete. Then, MABRA prompts for a value for the first field specified in the next record. After you make changes to all records within the constraint, MABRA responds with:

⋆ *UPDATE REQUEST NO.* **n** *FILED.*

MABRA then returns to:

⋆ *CONSTRAINTS*:

Enter a valid constraint (or enter *STOP* to return to the *PRINT FIELDS* prompt). If you enter a valid constraint, the *PRINT FIELDS* and the *SORT FIELDS* from the previous constraint are still in effect, in case you still need them. To reset these fields for a new constraint, enter *STOP* to return to the *PRINT FIELDS* prompt.

6. Deleting Records - *DR*

This command deletes records from the current system. Note that "deleted" records are not physically removed from the system, but are rather flagged (you could say, re-defined) as deleted. (See the *RB* command in the chapter, "Miscellaneous Commands" for an explanation about purging deleted records from MABRA.) When you enter *DR*, MABRA prompts with:

⋆ *DELETE RECORD(S)*
⋆ *CONSTRAINTS*:

Enter a valid MABRA constraint to identify the records you wish to delete (or enter *STOP* to return to the command prompt in Data Mode). MABRA responds with the message,

n *FOUND.*

to indicate that **n** records exist which meet the specified constraint. (If zero records are found, the *CONSTRAINTS* prompt is re-issued.) MABRA then issues:

⋆ *PRINT RECORDS TO BE DELETED? (Y/N)*:

This allows you to verify that the records identified for deletion are the correct ones. Enter *Y* to print the records; enter *N* to bypass printing. If you enter *Y*, MABRA prompts with:

** PRINT FIELDS:*

Specify which fields, within the located records, you wish to print (or enter *STOP* to return to the *CONSTRAINTS* prompt). The order in which you enter field names is the order in which they will print. This list can contain field names or certain field derivatives, separated by commas. See the Field Format table in the section, Listing Records - *LR*, in this chapter. If you enter valid print fields or derivatives, MABRA asks for the fields by which to sort the records:

** SORT FIELDS:*

Enter a list of sort fields (also as explained in the section, Listing Records - *LR*); enter a blank space and carriage return to indicate that no sorting is required; enter *STOP* to return to the *PRINT FIELDS* prompt.

If you request any print fields, MABRA responds with the following prompt to prepare you for output:

** ALIGN PAPER, TYPE SPACE, CR TO START.*

Set up the paper, enter a blank space and press carriage return to produce the listing (or enter *STOP* to return to the command prompt in Data Mode). Once the output finishes printing, again enter a blank space and press carriage return.

MABRA proceeds with the following prompt:

** FILE UPDATE REQUEST? (Y/N):*

Enter *Y* to file an update **request** for the deletions you just made. Note that changes are **not** made to the system until you enter the *UP* command (explained in the section, Initiating an Update Task - *UP*, in this chapter). If you enter *N*, the deletions are **not** made and MABRA responds with,

** NO RECORDS DELETED.*

and returns to the *CONSTRAINTS* prompt.

If you enter *Y*, MABRA prints the message,

** UPDATE REQUEST NO.* **nn** *FILED.*

and returns to the *CONSTRAINTS* prompt.

7. Batch Printing of Records - *BP*

This command retrieves specified records within a MABRA system and prints them on a highspeed printer. *BP* is identical to the *LR* command except that, with *BP*, a heading prints on the top of each page and a large number of records can be sorted and printed efficiently. When you enter *BP*, MABRA responds with:

** BATCH PRINT*
** CONSTRAINTS:*

Enter a valid MABRA constraint to identify the records you wish to print; enter *ALL* to print all records; enter *STOP* to return to the command prompt in Data Mode. If you enter valid constraints, MABRA responds with,

n *FOUND.*

indicating that **n** records exist which meet the constraints you specified. If zero records are found, the *CONSTRAINTS* prompt is re-issued so that you can enter different constraints. After records are found that meet your constraints, MABRA responds with:

** PRINT FIELDS:*

You should specify which fields, within the located records, you wish to print (or enter *STOP* to return to the *CONSTRAINTS* prompt). The order in which you enter the field names is the order in which they will print. The list can contain field names or certain field derivatives, separated by commas. See the Field Format table in the section, Listing Records - *LR*, in this chapter.

The number of characters required to print the specified fields of a record should not exceed 132. If more characters occur, a warning prints, saying that the overflow will wrap around to the next line. After you respond appropriately to the *PRINT FIELDS* prompt, MABRA prompts for the fields by which you wish to sort the records:

** SORT FIELDS:*

You can sort the located records by **any** combination of fields (not merely by *PRINT FIELDS*), with each field in either ascending or descending order. Enter a list of sort fields (as explained in the section, Listing Records - *LR*, in this chapter), if sorting is required; enter a blank space and press carriage return to indicate that no sorting is required; enter *STOP* to return to the *PRINT FIELDS* prompt.

After you respond appropriately to the *SORT FIELDS* prompt, MABRA prompts with:

** PRINT SPECS:*

Enter valid print specifications for HSPRINT, as described in the I.P. Sharp publication, SATN-8 (or enter *STOP* to return to the *CONSTRAINTS* prompt). After you enter print specifications, MABRA asks for delivery instructions as shown below.

** ENTER DELIVERY INSTRUCTIONS. TYPE 'STOP' TO END INSTRUCTIONS.*
:

The delivery instructions are for the APL operations staff, telling the operators how and where to deliver output. Enter as many lines of instructions as required; type *STOP* to end the input of instructions. If you enter *STOP* without giving any instructions, MABRA returns to the *CONSTRAINTS* prompt. If you enter *STOP* after you give your delivery instructions, MABRA prompts with:

** RUN AS AN N-TASK OR A B-TASK? (N/B):*

An N-task or a B-task processes your batch print job more cheaply than a T-task would, and leaves your terminal available for other work. An N-task runs immediately, whereas a B-task runs when system resources can easily accommodate it. Because of its deferred execution, a B-task costs less than an N-task. For more information about N-tasks and B-tasks, see the I.P. Sharp publications, "Batch Tasks in SHARP APL", for a general explanation, and see "SATN-4" and "SATN-5", for a more technical explanation.

Enter *N* or *B*, depending on the urgency and the affordability of your application (or enter *STOP* to return to the *CONSTRAINTS* prompt). If you enter *N* or *B*, MABRA issues the following prompt:

```
* SUBMIT REQUEST? (Y/N):
```

Enter *Y* to initiate the *BP* N-task or B-task; enter *N* or *STOP* to return to the *CONSTRAINTS* prompt. If you enter *Y*, MABRA responds with,

```
* PRINT FILE IS: Pmmddhhmmss
* B-TASK REQ. NO. n
      or
* N-TASK n
```

where **mmddhhmmss** represents the month, day, hour, minute, and second of the task submission, and **n** represents the B-task or N-task number of your *BP* (HSPRINT) job.

MABRA then returns to the *CONSTRAINTS* prompt.

If the records you wish to batch print require sorting, the *BP* N-task or B-task creates two **temporary** files: *SORTIN***xx** and *SORTOUT***xx** (where the **xx**'s represent numbers that uniquely identify your sort files). Sorting is done by the SORTREQ utility which resides in workspace 1 *FILESORT*. If *BP* runs as an N-task, MABRA submits a sort request and, because of an N-task's high priority, the N-task waits—until the SORTREQ completes—before continuing. If *BP* runs as a B-task, MABRA submits a sort request and, because of a B-task's low priority, the B-task terminates and MABRA submits a second B-task to start **after** the sort completes.

After a *BP* job completes, when you re-enter the MABRA system or use the *NS* (new snapshot) command, the status of the *BP* job displays regarding the job's: N-task number or B-task number(s), the SORTREQ number (if applicable), the HSPRINT request number, and the time of job completion.

8. Initiating an Update Task - *UP*

The commands *AR*, *CR*, *CD*, and *DR* do not actually modify a MABRA system but rather **request** that the system be modified. The *UP* command initiates an N-task or a B-task to **process** all such pending requests. For more information about N-tasks and B-tasks, see the I.P. Sharp publications, "Batch Tasks in SHARP APL", for a general explanation, and see "SATN-4" and "SATN-5", for a more technical explanation.

MABRA uses a separate command for performing updates to records for three reasons. By collecting many records into one task, MABRA reduces system costs. By looking at an accumulation of tasks, MABRA lets you (or your supervisor) evaluate progress in your work. By allowing a separate procedure to perform updates, MABRA lets you withdraw an update task—if you realize you made a mistake—before the task processes.

When you enter *UP*, MABRA responds with,

```
* INITIATE UPDATE TASK
* x REQUESTS PENDING UPDATE.
```

where **x** is the number of requests waiting to be updated. If **x** is zero, there are no outstanding requests to be updated. (To check whether an update task has completed, execute *UP*. If the result is 0, this indicates that any pending update has completed.)

MABRA next issues the prompt:

```
* RUN AS AN N-TASK OR B-TASK? (N/B):
```

Enter `N` or `B`, depending on the urgency and the affordability of your application; enter `STOP` to bypass the request for an update task and to return to the command prompt in Data Mode. If you enter `N` or `B`, MABRA responds with,

```
* x-TASK NO. n
```

to confirm that the update task has been initiated. **x** is either `N` (N-task) or `B` (B-task), and **n** is either the task-ID of the N-task or the request number of the B-task that has been submitted.

Processed update requests exist, one per component, in MABRA's `UPD` file (see the chapter, "File Documentation"). After 50 such file components accumulate, MABRA gives you the opportunity, upon entering Data Mode, to purge these processed update requests. If you choose not to purge these requests, MABRA increases the `UPD` file limit by another 50 components.

Important Note: The update is run as an N-task or as a B-task to save on system costs and to keep your terminal available to you. Not being a T-task, information about job completion and about record additions and deletions is not immediately available to you. Because you may be functioning in Data Mode at the same time that an N-task or a B-task is processing your update request, a method exists for keeping your Data Mode session separate from possible updates-in-progress.

To enable Data Mode consistency throughout an updating task, MABRA takes a "snapshot" of the record environment when you enter Data Mode. Although you can add, change, and delete records when you are in Data Mode, only the **changes** to a record will be visible to you. The initial snapshot **censors** the updating of **additions** and **deletions** while you remain in Data Mode. This snapshot does not change until you either enter the `NS` command to take a new snapshot, or until you leave Data Mode and then re-enter it.

In other words, if an update task runs while you remain working in Data Mode and if you don't execute `NS`, you will **not** see added records and will **still** see deleted records. Otherwise, if you leave and re-enter Data Mode or if you execute `NS`, additions and deletions to the system will be visible to you.

9. Taking a New Snapshot - `NS`

MABRA takes a snapshot of all records in the system whenever you enter Data Mode. As mentioned at `UP`, this snapshot censors the addition and deletion of records during update routines so that you are not confused by updates-in-progress. The process of **re-entering** Data Mode in order to guarantee the latest version of the system is slightly expensive and time-consuming. The `NS` command bypasses such unnecessary expense and loss of time by letting you take a new snapshot **within** Data Mode. When you enter `NS`, MABRA responds with,

```
* SNAPSHOT TAKEN.
```

to show that you now have the latest version of Data Mode.

Usually you want a new snapshot because you recently initiated an update task and now wish to see the system changes. When you enter `NS` after initiating an update task MABRA returns with,

```
* UP x-TASK COMPLETED AT: hh:mm:ss dd/mm/yy
* SNAPSHOT TAKEN.
```

if the update task has completed. The **x** indicates either an N-task or a B-task; **hh:mm:ss dd/mm/yy** represents the time and day of completion.

10. Withdrawing an Update Request - *WU*

This command withdraws pending update requests from the system. When you enter *WU*, MABRA prompts with:

```
* WITHDRAW UPDATE REQUESTS
* WHICH NUMBER(S)?:
```

Enter a list of the update request numbers, separated by blanks, that are to be withdrawn (or enter *STOP* to return to the command prompt in Data Mode). MABRA then prints the message,

```
* DONE.
```

to confirm the withdrawal of the update requests. MABRA then returns to the *WHICH NUMBER(S)* prompt.

Note that a request to update the system with the *UP* command (*AR*, *CR*, *CD*, *DR*) is **nullified** by executing the *WU* command. However, *UP* requests which are withdrawn by the *WU* command **remain on file** to provide an audit trail of *UP* requests. Therefore, do not be confused to find *UP* requests pending after you withdrew them with the *WU* command. To eliminate the *UP* requests that are still pending after a *WU* request, you merely need to run the *UP* command to effectively process (withdraw) them. Read about the *LU* command, below, to learn how MABRA lists all update requests.

11. Listing an Update Request - *LU*

This command lists information regarding update requests which are either pending or completed. When you enter *LU*, MABRA responds with:

```
* LIST UPDATE REQUESTS
* COMPLETED UPDATE REQUEST NUMBERS: a → b
* PENDING UPDATE REQUEST NUMBERS: x → y
* ENTER UPDATE REQUEST NUMBER RANGE:
```

a and **b** represent the lower and upper numbers, respectively, in the range of update requests which have been processed by the *UP* command; **x** and **y** represent the lower and upper numbers in the range of update requests waiting to be processed by *UP*. Specify the **range** of update requests that you wish to list, by first entering the low update request number and then, the high number (or enter *STOP* to return to the command prompt in Data Mode). If you enter only one number, that specific request will display. If you enter a valid request range, MABRA then issues the following message to prepare you for output:

```
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

Set up the paper, enter a blank space and press carriage return to start printing your desired update requests. After the list finishes printing, again enter a blank space and press carriage return to get back to the *ENTER UPDATE REQUEST NUMBER RANGE* prompt.

Note that when you list update requests, the number of *COMPLETED UPDATE REQUESTS* includes *UP* tasks which you withdrew by using *WU*. However, when you proceed to display a range of *UPDATE REQUEST NUMBERS*, all *UP* tasks that you withdrew are labelled as such, showing that no updates occurred in those cases.

CROSSTABULATION ANALYSIS

Crosstabulation is the evaluation of the frequency of a given value for a field within a given sample of records. A field has a finite set of values. For example, consider a personnel system with the following fields:

Field Name	**Description of Field's Value**	**Number of Values**
NAME	surname	equals total records
SEX	sex (M or F)	2
LOCATION	location (CT, CM, EL, or EC)	4
DOBIRTH	birthdate	equals total records
DOJOIN	date person joined the company (to derive length of service)	equals total records
SALARY	salary	equals total records

CT, CM, EL, and EC are *LOCATION* codes for Toronto Canada, Montreal Canada, London England, and Coventry England, respectively.

Note that within this system the field name *SEX* can have one of only two values for a field, and *LOCATION* can have one of only four values. *NAME*, *DOBIRTH*, *DOJOIN*, and *SALARY* can have as many values as there are records in MABRA.

Often you are not interested in all field values when performing a crosstabulation analysis, but rather, in a **range** of values. For example, the salaries of interest can be selected from *SALARY* by using such a range (e.g., less than 15000; 15000 to 25000; greater than 25000).

For the sample system defined above, a typical crosstabulation of *SEX* and *LOCATION* might appear as shown below. The left margin indicates rows of information containing *SEX* distribution; the top heading indicates columns of information by *LOCATION*.

	TORONTO	*MONTREAL*	*LONDON*	*COVENTRY*	*TOTAL*
MALE	300	200	50	50	600
FEMALE	200	150	50	0	400
TOTAL	500	350	100	50	1000

An age (α*DOBIRTH*) and *SALARY* crosstabulation might appear as follows. The left margin indicates rows of information based on ranges in age; the top heading indicates columns of information by ranges in *SALARY*.

	[1]	[2]	[3]	*TOTAL*
UP TO 25 *YEARS*	300	200	50	550
25 *TO* 35 *YEARS*	600	800	300	1700
35 *YEARS AND OVER*	200	300	400	900
TOTAL	1100	1300	750	3150

The *SALARY* ranges are:

[1] salary is less than 15,000
[2] salary is between 15,000 and 25,000
[3] salary is greater than 25,000

MABRA allows multi-dimensional crosstabulations (e.g., age by length of service by location), as well as field totalling and averaging (e.g., average salary in certain age ranges).

Crosstabulation Parameters

For each crosstabulation, you must provide MABRA with specific information, as outlined below.

1. When requesting a crosstabulation analysis, you first specify what **type** of analysis you want. You can select any one (or any combination) of the following three analysis options:

    - frequency counting (e.g., number in each location)
    - totalling (e.g., total salary by location)
    - averaging (e.g., average age in each salary range)

2. You must specify the **group of records** for analysis by entering a MABRA constraint. (See the section about MABRA Constraints, in the chapter, "Data Mode".)

3. You must specify a **list of field names (or certain derivatives)**, delimited by commas. The formats are:

    field-name - if you apply the analysis to the whole field

    αdate-field-name - if you apply the analysis to the number of years from today for a date field (e.g., `αDOJOIN` being the length of service from today)

    field-name[index] - if you apply the analysis to a section of the field (e.g., `LOCATION[1]` being the country identifier)

    The following are examples of valid lists of field names and/or their derivatives:

    `SEX,LOCATION` - sex by location
    `αDOBIRTH,SALARY` - age by salary
    `NAME[1]` - first letter of name
    `LOC[1],SEX,αDOJOIN` - country by sex by length of service

4. If MABRA can analyze the data in a field more efficiently by apportioning the data, you should specify **range points** in order to limit the points of comparison. For example, you can enter range points of "15,000" and "25,000" for the *SALARY* field. MABRA then takes these range points and puts them into the following ranges:

    ```
    SALARY<15000
    15000≤SALARY<25000
    25000≤SALARY
    ```

Running a Crosstabulation Analysis - *AN*

To run a crosstabulation analysis, you must use the *AN* (analysis) command. MABRA prompts for all necessary parameters. When you enter *AN*, MABRA responds with:

```
* CROSSTABULATION ANALYSIS
* OPTION? (F/T/A/FT/FA/FTA/TA):
```

You should specify one of the following options (or enter *STOP* to return to the command prompt in Data Mode).

F	frequency counting
T	field totalling
A	field averaging
FT	frequency counting and field totalling
FA	frequency counting and field averaging
FTA	frequency counting, field totalling, and field averaging
TA	field totalling and field averaging

When you specify an appropriate option, MABRA prompts with:

```
* CONSTRAINTS:
```

Enter a valid MABRA constraint to specify the group of records for analysis (or enter *STOP* to return to the *OPTION* prompt). If you specify a valid constraint, MABRA prompts with:

```
* ENTER LIST OF FIELD NAMES:
```

Enter a list of the field names, separated by commas, for analysis (or enter *STOP* to return to the *OPTION* prompt). If you specify a list of field names, MABRA prompts with:

```
* USE RANGES? (Y/N):
```

Enter *Y* if you require range points for any field names; enter *N* if you do not require range points; enter *STOP* to return to the *OPTION* prompt.

If you specify *Y*, MABRA lists the field names that you specified and asks which field names require range points.

** FIELD NAMES ARE:*
1. **field-name-1**
2. **field-name-2**

↓
↓

n. **field-name-n**

```
* ENTER NUMBER OF FIELD NAME:
```

Enter the number of the field name for which you wish to specify range points (or enter *STOP* to end the input of ranges). If you enter the line number of a field's name (as shown at the left, in the above example), MABRA responds with,

* **field-name-n**
:

where **n** is the number you specified in response to the *ENTER NUMBER OF FIELD NAME* prompt. Now enter each range point, typing one per line. Enter *STOP* after the command prompt (:) to indicate that all range points have been specified for that field name.

If the analysis option you specified in response to the *OPTION* prompt contains a *T* (totals) or an *A* (averages), MABRA issues:

* *FIELD TO BE TOTALLED*:

Enter **field-name**, or α**date-field-name**; otherwise, enter *STOP* to return to the *OPTION* prompt.

If the option you specified in response to the *OPTION* prompt contained a *T* or an *A* **and** the field to be totalled has a type of *REAL*, MABRA prompts with:

> * *ROUND TO HOW MANY DECIMAL PLACES?*
>
> Enter an integer between ¯10 and 10, specifying the number of decimal places to round the averages. For example:
>
> 0 rounds to the nearest whole number
> 2 rounds to the nearest 1/100th
> ¯3 rounds to the nearest 1000

After you specify all the necessary parameters in the manner described above, MABRA prompts with:

* *DO ANALYSIS? (Y/N)*:

Enter *Y* to process the crosstabulation; enter *N* to cancel the analysis and to return to the *OPTION* prompt.

If you enter *Y*, MABRA prints the following to prepare you for output:

* *ALIGN PAPER, TYPE SPACE, CR TO START*.

Set up your paper, enter a blank space and press carriage return to produce the crosstabulation listing of field values and/or their derivatives (or enter *STOP* to return to the command prompt in Data Mode).

CODED FIELDS

MABRA allows you to specify a code definition (**coded** or **un-coded**) for the values of a particular field. This means that the values you enter for a **coded** field are coded or abbreviated versions of their full descriptions; values for an **un-coded** field are literal, having no descriptions. For example, a coded field called *LOC*, which stores an employee's geographical **location**, could contain the following values: *CM*, *CV*, *EB*, *EM*, *FP*, and *UD*. These values may be tabulated as:

Code **Description**

CM - MONTREAL, CANADA
CV - VANCOUVER, CANADA
EB - BIRMINGHAM, ENGLAND
EM - MANCHESTER, ENGLAND
FP - PARIS, FRANCE
UD - DALLAS, U.S.A.

What you have above is the **code-description table** for the coded field, *LOC*. By definition, this table comprises multiple rows that contain a code value (numeric or character) and its corresponding description. A code-description table resides in a MABRA file (see the *DIR* file, described in the chapter, "File Documentation") and is created, referenced, and maintained by the MABRA commands discussed below.

In the above table, the first character of *LOC*'s code represents a country identifier, and the second character, a city identifier. By coding the *LOC* field in this way, you simplify the identification of all employees in a particular city or country when using a MABRA constraint. For example, the constraint *LOC*[1]=*C* identifies all employees working in Canada and *LOC=EB* identifies all employees working in Birmingham, England.

You save space by storing these location codes rather than the field's full location values. The code-description table is helpful because it enables you to **reference** codes' descriptions if you are uncertain of them and to **substitute** descriptions for codes when you print reports.

If you wish to display the descriptions of coded values (using *LR*, *CR*, *DR*, or *BP*), you can do so by enclosing the name of the field **within parentheses**. If, however, a blank "description" returns for some coded values, this means that when you added or changed records (using *AR*, *CR*, or *CD*) you neglected to add their descriptions (using *AC*, described below) to the code-description table.

You must use the following commands to create, reference, or maintain a code-description table for a coded field:

CF - change a field's (code) definition
AC - add codes to a field's code-description table
LC - list a field's code-description table
DC - delete a code from a field's code-description table

1. Changing the Code Definition of a Field - *CF*

The section, Changing Fields - *CF*, in the chapter, "Data Mode", explains that *CF* changes all field definitions (*NAME*, *WIDTH*, *INVERT*, *CODE*, *⎕FMT*) except a field's type. However, the way to change a field's **code** definition is not explained in "Data Mode". The process is explained separately, in this chapter, in order to highlight its detail.

When you enter *CF*, MABRA responds with:

```
* CHANGE FIELD
* FIELD NAME:
```

Enter the name of the field whose code definition you wish to change (or enter *STOP* to return to the command prompt in Data Mode). When you enter a field's name, MABRA responds with:

```
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT:
```

Enter *CODE* to continue (or enter *STOP* to return to the *FIELD NAME* prompt).

If the code definition of the field you enter is "un-coded", MABRA prompts with:

```
    * FIELD'S CODE DEFINITION IS 'UN-CODED'.
    * CHANGE TO 'CODED'? (Y/N):
```

Enter *Y* to change the field's definition to "coded"; enter *N* or *STOP* to return to the *CHANGE NAME, WIDTH* . . . prompt; enter *STOP* to return to the *CHANGE NAME, WIDTH* . . . prompt. If you enter *Y*, MABRA prompts with:

```
    * WIDTH OF CODES' DESCRIPTIONS:
```

Enter a positive integer representing the number of characters that accommodate a code's description. This number should be as accurate as possible, since the maximum allowable width for all descriptions is based on your response. After you specify the width of the codes' descriptions, MABRA calculates the size of the code-description table:

```
    * SUGGESTED MAXIMUM NUMBER OF CODES IS n.
    * IS n OK? (Y/N):
```

n represents the greatest number of rows (one code and its description per row) that MABRA can accommodate in the code-description table. MABRA derives this number by calculating available space after learning the descriptions' width (from your response to the *WIDTH OF CODES' DESCRIPTIONS* prompt). Enter *Y* to set up the **empty** code-description table; enter *N* to request a different width; enter *STOP* to return to the *CHANGE NAME, WIDTH*. . . prompt.

If you enter *Y*, MABRA indicates that the specified field now has a coded definition, and prints the message:

```
    * CODED.
```

MABRA then returns to the *CHANGE NAME, WIDTH*. . . prompt.

If the code definition of the field you enter is already "coded", MABRA responds with:

```
    * FIELD DEFINITION IS 'CODED'.
    * UN-CODE FIELD OR CHANGE WIDTH OF CODES' DESCRIPTIONS? (U/W):
```

Enter *U* to change the code definition to "un-coded" and to delete the field's code-description table; enter *W* to change the width of the codes' descriptions; enter *STOP* to return to the *CHANGE NAME, WIDTH*. . . prompt.

If you enter *U*, MABRA responds with:

```
* UN-CODED.
```

MABRA then returns to the *CHANGE NAME, WIDTH...* prompt.

If you enter *W*, MABRA prompts with:

```
* NEW WIDTH OF CODES' DESCRIPTIONS:
```

Enter a positive integer that accommodates the width of a code's description (or enter *STOP* to return to the *CHANGE NAME, WIDTH...* prompt). If you enter a number that is less than the "current" width, MABRA truncates the descriptions along the field's right boundary. If the number is greater than the "current" width, MABRA pads new descriptions with blanks along the field's right boundary.

MABRA re-calculates the size of the code-description table and responds with:

```
* SUGGESTED MAXIMUM NUMBER OF CODES IS n.
* IS n OK? (Y/N):
```

n is the new maximum number of entries suggested for the code-description table. You may exceed this number, but a *WS FULL* error may occur as a result. Enter *Y* to confirm that the new width for the code-description table is acceptable; enter *N* to request a different width; enter *STOP* to return to the *UN-CODE FIELD OR CHANGE WIDTH...* prompt. If you enter *Y*, MABRA prints the following to confirm the change in width:

```
* WIDTH OF CODES' DESCRIPTIONS IS CHANGED.
```

MABRA then returns to the *CHANGE NAME, WIDTH...* prompt.

2. Adding a Code to the Code-Description Table - *AC*

This command adds a code and its respective description to the code-description table of a specified coded field. When you enter *AC*, MABRA responds with:

```
* ADD CODES/DESCS.
* FOR WHICH FIELD:
```

Enter the name of the field for which you wish to add a code (or enter *STOP* to return to the command prompt in Data Mode). If you specify a field whose code definition is "not coded", an error message results. If you specify a valid field name, the system prints the message,

```
* CURRENT NUMBER OF CODES IS x.
* SUGGESTED MAXIMUM NUMBER OF CODES IS y.
```

as a reminder of the current size of the code-description table for the specified field. MABRA then prompts for your additional codes and their corresponding descriptions:

```
* CODE:
```

Enter one new code name here (or enter *STOP* to return to the *FOR WHICH FIELD* prompt).

```
* DESC.:
```

Enter a description to correspond with the code given above (or enter `STOP` to retract the above code name and to return to the `CODE` prompt).

After you enter a description to the `DESC.` prompt, MABRA re-prompts for the next code and its corresponding description. Note that if you specify a code which already exists, the description you subsequently enter **replaces** the old description. When you finish entering all codes and descriptions, type `STOP` to update the table. MABRA then prints the following message to confirm that the code-description table has been updated:

```
* UPDATED.
```

MABRA then returns to the `FOR WHICH FIELD` prompt.

3. Listing a Code-Description Table - `LC`

The `LC` command lists a code-description table in either code-ascending order or description-ascending order. When you enter `LC`, MABRA prompts with:

```
* LIST CODES/DESCS.
* FOR WHICH FIELD:
```

Enter the name of the field whose code-description table you wish to list (or enter `STOP` to return to the command prompt in Data Mode). MABRA responds with:

```
* SORT BY CODE OR DESCRIPTION? (C/D):
```

Enter `C` to list the table in code-ascending order; enter `D` to list the table in description-ascending order; enter `STOP` to return to the `FOR WHICH FIELD` prompt. If you enter `C` or `D`, the system prints the follwoing message to prepare you for output:

```
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

Set up your paper, enter a blank space and press carriage return to start the listing. Before the table prints, however, MABRA prints the following information, regarding the number of existing codes and the upper limit for the number of codes in this field:

```
* CURRENT NUMBER OF CODES IS x.
* SUGGESTED MAXIMUM NUMBER OF CODES IS y.
```

When the output finishes printing, again enter a space and hit carriage return to get back to the `FOR WHICH FIELD` prompt.

4. Deleting a Code from the Code-Description Table - `DC`

The `DC` command removes a code and its respective description from a code-description table. When you enter `DC`, MABRA prompts with:

```
* DELETE CODES/DESCS.
* FOR WHICH FIELD:
```

Enter the name of the field whose code-description table requires deletions (or enter *STOP* to return to the command prompt in Data Mode). When you enter a valid field name, MABRA responds with:

```
* CODE:
```

Enter the code that you wish to remove. Note that the *CODE* prompt issues continually until you enter *STOP*. After you enter *STOP*, MABRA prints the message,

```
* UPDATED.
```

to confirm that the specified codes and their descriptions are deleted from the code-description table. MABRA then returns to the *FOR WHICH FIELD* prompt.

MISCELLANEOUS COMMANDS

This chapter explains five miscellaneous commands. Two of these commands, *RB* and *RS*, deal with setting new **size limits** for a system's master file. The third command, *CS*, enables you to **copy** a MABRA system to which you have access. The fourth command, *FO*, lets you control the **field order** for input during the *AR* (add record) command. The remaining command, *ST*, displays information about the physical **status** of a MABRA system.

1. Reblocking the Master File - *RB*

The *RB* command changes the master file's blocking factor for the current system. The master file contains a system's records of data. Each record in a MABRA system has the same size, referred to as the **record size**. The **blocking factor** is a value that indicates the number of records that MABRA stores in a file component (or **block** of data). MABRA calculates the **block size** of the master file by multiplying the record size (in bytes) by the blocking factor. The blocking factor of this file is set when MABRA **creates** a system, and is based on a default, average record size.

Given a workspace size of 256 kilobytes (256000 bytes), the size of one block in the master file should be about 20 kilobytes. For example, in a system with an average record size of 100 bytes, MABRA sets the blocking factor to 200 records per block: `20000÷100=200`.

Such a blocking factor may need resetting in the following cases:

- Your record size has exceeded 100 bytes. This condition could prevent you from adding any more fields to the system. To avoid this condition, a blocking factor smaller than 200 records is recommended.

- Your record size is much less than 100 bytes and may never increase. In this case, a blocking factor larger than 200 records is recommended in order to increase the efficiency of searches and retrievals.

If you know all the fields that you require for a particular system, prior to the creation of this system, you can calculate the record size by summing the space required for each field in the system:

Field Type	**Bytes Used Per Element**
CHAR	1
INTE	4
REAL	8
BINA	.125
DATE	4

For example, a system with the following fields:

NAME - 15 characters
SALARY - 1 integer
DOBIRTH - 1 date

would have the following field sizes:

NAME - 15 × 1 bytes
SALARY - 1 × 4 bytes
DOBIRTH - 1 × 4 bytes

The total number of bytes required equals 23. Therefore, an ideal blocking factor would be represented as: `⌈20000÷23`, equalling 870.

When you enter the *RB* command, MABRA responds with,

```
* REBLOCK MASTER FILE
* NUMBER OF RECORDS (INCLUDING DELETED)=x
* NUMBER OF DELETED RECORDS=y
* PURGE DELETED RECORDS? (Y/N):
```

where **x** is the total number of records in the system, and **y** is the number of records you have deleted previously with *DR*. At this point, these deleted records can be physically removed from the system in order to clean up the system and reduce file space. Enter *Y* to remove the deleted records; enter *N* to keep them; enter *STOP* to return to the command prompt in Data Mode. MABRA then prompts with,

```
* CALCULATED BLOCKING FACTOR=n, DO YOU WISH TO INPUT YOUR OWN? (Y/N):
```

where **n** is the calculated optimum blocking factor, based on the size of existing fields. Enter *Y* to specify that you wish to enter your own blocking factor; enter *N* to keep the calculated blocking factor; enter *STOP* to return to the *PURGE DELETED RECORDS* prompt. If you specify *Y*, MABRA prompts with,

```
* ENTER BLOCKING FACTOR (≤n):
```

where **n** is the calculated blocking factor. Enter an integer less than or equal to **n** (or enter *STOP* to return to the *CALCULATED BLOCKING FACTOR* prompt). If you enter a valid integer, MABRA reblocks the master file, and prints the message,

```
* REBLOCKING DONE.
```

to confirm the new blocking factor. MABRA then returns to the command prompt in Data Mode.

2. Resizing the Master File - *RS*

You, the *CREATOR*, set the limit for the maximum number of records for a MABRA system when you first create the system. You set a record limit based on your estimate of the number of records you will enter into the system. If you reach this limit you can increase it by using the *RS* command to resize the master file. When you enter *RS*, MABRA prompts with,

```
* RESIZE SYSTEM
* ENTER MAXIMUM NUMBER OF RECORDS (≥n):
```

where **n** is the total number of records in the current system (including deleted records). Enter a new limit (or enter *STOP* to return to the command prompt in Data Mode). If you enter a valid integer, MABRA sets the record limit to that value, and prints the following message to confirm it:

```
* RESIZING DONE.
```

MABRA then returns to the command prompt in Data Mode.

3. Copying a MABRA System - *CS*

You might wish to copy another system, or a subset of it, for the purposes of training someone, demonstrating the system, using it as a model, or merging two compatible systems. The *CS* command copies another MABRA system into your current system.

The system that you wish to copy is called the *SOURCE* system. The system that you are within, and that you wish to copy into, is called the *TARGET* system. You must have valid access to copy the *SOURCE* system if you are not the *CREATOR* (see the chapter, "Multi-User MABRA").

When you enter *CS*, MABRA prompts with:

```
* COPY SYSTEM
* ENTER SOURCE CREATOR AND SYSTEM NUMBER:
```

Enter the *CREATOR*'s user number and the system number of the *CREATOR*'s system that you wish to copy (or enter *STOP* to return to the command prompt in Data Mode). If you have access to this *SOURCE* system, MABRA responds with:

```
* ENTER RECORDS TO BE COPIED, IN THE FORM OF A CONSTRAINT
* CONSTRAINTS:
```

Enter a valid MABRA constraint to identify the records you wish to copy from the *SOURCE* system; enter a blank space and press carriage return to copy the *SOURCE* system's field definitions only; enter *STOP* to return to the *ENTER SOURCE CREATOR* prompt.

MABRA then prompts with:

```
* ENTER FIELDS TO BE COPIED.
* FIELDS:
```

Enter a list of field names, separated by commas, to identify those fields you wish to copy from the *SOURCE* system; enter *ALL* to copy all fields; enter *STOP* to return to the *ENTER SOURCE CREATOR...* command.

If you enter a valid list of field names, MABRA prompts with,

```
* MORE FIELDS? (Y/N):
```

to enable you to identify more fields. Enter *Y* if you wish to enter more field names; enter *N* to signal that your list of field names is already complete; enter *STOP* to return to the *ENTER SOURCE CREATOR...* prompt.

If you respond to the *MORE FIELDS* prompt with *N*, MABRA prompts with:

```
* DO YOU WISH TO OVERWRITE THE TARGET SYSTEM? (Y/N):
```

If you enter *Y*, MABRA erases the fields and records in the *TARGET* system and replaces them with the fields and records from the *SOURCE* system.

If you enter *N*, MABRA combines the *TARGET* and the *SOURCE* systems by appending the *SOURCE* records to the bottom of the *TARGET* system. Certain field definitions in the *SOURCE* system must be compatible with the *TARGET* system before the *SOURCE* records can be appended. These *SOURCE* field definitions are a field's **name**, **type**, and **number of elements**.

If you enter *Y* **or** *N* to the above prompt, MABRA first responds with the following, before proceeding with your command:

```
* ARE YOU SURE? (Y/N):
```

Enter *Y* to continue; enter *N* to return to the *ENTER SOURCE CREATOR...* prompt.

If you responded to the *DO YOU WISH TO OVERWRITE...* prompt with *Y*, MABRA responds with,

```
* ENTER MAXIMUM NUMBER OF RECORDS EXPECTED IN TARGET SYSTEM (≥n):
```

where **n** represents the current number of records in the *TARGET* system. If you respond with a valid integer, MABRA returns with:

```
* SYSTEM COPIED.
```

After MABRA copies the *SOURCE* system, MABRA returns to the command prompt in Data Mode.

If you responded to the *DO YOU WISH TO OVERWRITE...* prompt with *N*, **and** your *SOURCE* and *TARGET* field definitions are compatible, MABRA responds with:

```
* SYSTEM COPIED.
```

4. Field Ordering for *AR - FO*

The *FO* command **arranges the field prompts** for the *AR* (add record) command into whatever alternative orders you specify. You may need to use this field ordering in the following circumstances:

- You may wish to **suppress** certain field prompts. Maybe data for some fields in your system is available for input only at a certain time. When you are adding records at other times, you can specify a field order that prompts only for the fields for which you have data. Depending on the data type, MABRA assigns values of blanks or zeros to suppressed fields. Note that when using a field order that suppresses fields, you cannot use the "jump to" (→) feature to access a suppressed field.

- You may need **various** field orders, depending on the format of your information source. Your system may take its field data from several different standardized forms. You can specify a different field order for each form so that the field prompts occur in the same order as the information on the form.

By default, *AR* already has a field order—the order that results when you use *LF* to list fields. By using *FO*, you can assign multiple field orders—a different order to meet each different input situation.

When you enter *FO*, MABRA responds with:

```
* AR FIELD ORDER
* THERE ARE ALREADY n AR FIELD ORDERS.
```

where **n** represents the number of field orders currently in the system. Initially, **n** is 1, indicating the default *LF* field order.

MABRA prompts with the following options for field ordering:

```
* ADD, CHANGE, DELETE, OR LIST? (A/C/D/L):
```

Enter the initial of one of the above options (or enter *STOP* to return to the command prompt in Data Mode).

a. If you specify *A*, MABRA responds with,

```
* ADDING FIELD ORDER n
* FIELDS:
```

where **n** represents the identification number of the field order you are about to add.

At the *FIELDS* prompt, enter your choice of field names, separated by commas, in whatever sequence you desire for this field order (or enter *STOP* to return to the *ADD, CHANGE, DELETE, LIST...* prompt). If you enter a list of field names, MABRA prompts with:

```
* MORE? (Y/N):
```

Enter *Y* if you wish to add more field names; enter *N*—if your field names are already complete—to add this new field order to the system; enter *STOP* to return to the *ADD, CHANGE, DELETE, LIST...* prompt.

b. If you specify *C*, MABRA responds with:

```
* CHANGE WHICH FIELD ORDER:
```

Enter the identification number of the field order you wish to change (or enter *STOP* to return to the *ADD, CHANGE, DELETE, LIST...* prompt). If you enter a number, MABRA prompts with:

```
* FIELDS:
```

Enter your choice of field names in the sequence you desire for this field order (or enter *STOP* to return to the *ADD, CHANGE, DELETE, LIST...* prompt). If you enter a list of field names, MABRA issues:

```
* MORE? (Y/N):
```

Enter *Y* if you wish to add more field names; enter *N*—if your field names are already complete—to change this field order in the system; enter *STOP* to return to the *ADD, CHANGE, DELETE, LIST...* prompt.

c. If you specify *D*, MABRA returns with:

```
* DELETE WHICH FIELD ORDERS:
```

Enter the identification numbers—separated by blanks—of the field orders that you wish to delete (or enter *STOP* to return to the *ADD, CHANGE, DELETE, LIST...* prompt). After you specify one or more numbers, MABRA responds with,

```
* DELETED.
```

to confirm the deletion of the specified field orders. MABRA then returns to the *DELETE WHICH FIELD ORDERS* prompt.

d. If you specify *L*, MABRA returns with:

```
* LIST WHICH FIELD ORDERS:
```

To list certain field orders, enter their field order identification numbers, separated by blanks; enter *ALL* to list all field orders; enter *STOP* to return to the *ADD, CHANGE, DELETE, LIST...* prompt. If you enter a list of identification numbers or *ALL*, MABRA responds with the following, to prepare you for output:

```
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

Set up your paper for output, enter a blank space and press carriage return to list the field orders at your terminal. After your output prints, again enter a blank space and press carriage return to get back to the *ADD, CHANGE, DELETE, LIST...* prompt.

After you use *FO* to add field orders to the *AR* command, a new prompt regarding field ordering occurs when you enter *AR*:

```
nn:AR
* ADD RECORD
* FIELD ORDER:
```

After the last prompt, you should enter the identification number of the field order you want for adding record input (or enter *STOP* to return to the command prompt in Data Mode).

5. Displaying the Status of a MABRA System - *ST*

The *ST* command displays the physical status of a MABRA system. If you enter *ST*, MABRA responds with the following information:

```
TOTAL NUMBER OF RECORDS: n
NUMBER OF DELETED RECORDS: n
MAXIMUM NUMBER OF RECORDS EXPECTED: n
RECORD SIZE: n (BYTES)
MASTER FILE'S BLOCKING FACTOR: n
NUMBER OF FIELDS: n
NUMBER OF INVERTED FIELDS: n
NUMBER OF CODE/DESC. TABLES: n
FILE STATUS
    file-name: n (BYTES)              (the sizes of 7 MABRA files display here)

NUMBER OF PENDING UPDATES: n
FIRST PENDING UPDATE REQUEST NUMBER: n
NUMBER OF COMPLETED UPDATES: n
```

```
TOTAL NUMBER OF UPDATES: n
THE FOLLOWING USERS HAVE ACCESS TO THE SYSTEM: user-number(s)
ACTIVE USERS: user-number(s)
```

MULTI-USER MABRA

By default, the *CREATOR* is the only user who has access to any MABRA systems on his or her user number. If another user number makes an unauthorized attempt to access any such systems, one of the following error messages results:

? NO SYSTEMS EXIST ON **creator's-user-number**.

? SYSTEM DOES NOT EXIST.

It is quite appropriate for the *CREATOR* to allow other users some level of access to a MABRA system, say, for updating or reporting purposes. For example, in a personnel system for a multi-national company, it may be necessary for each office in that company to be able to update and report on all its own information, as well as to report on selected information related to other offices.

There are three different levels of access that the *CREATOR* can grant to other users: command access, record access, and field access.

1. With **command access**, you, the *CREATOR*, can give other users access to any combination of the following commands:

    LF - list all fields' definitions
    AR - add new records
    CR - change records having the same value in a field
    CD - change records having different values in a field
    DR - delete records
    LR - list records
    BP - batch print
    AN - crosstabulation analysis
    LC - list codes
    UP - initiate an update task
    WU - withdraw an update transaction
    LU - list an update transaction
    NS - take a new snapshot
    CS - copy a system

2. With **record access** you can give other users access to any record or group of records that are expressed as a constraint. Note that the **combinations** of **command access** and **field access** which you specify for a given user will control what the user may do to the accessible records.
3. With **field access** you can give other users the ability to add, change, or list values for certain fields within a record. Field access is broken down as follows:

AR field access -	determines the fields which specified users can add to a record when using the *AR* command.
CR field access -	determines the fields which specified users can change within a record when using the *CR* and *CD* commands.
LR field access -	determines the fields which specified users can list within a record when using the *LR*, *BP*, *CR*, *CD*, *DR*, and *AN* commands.

You must use the following commands to control the command access, record access, and field access that you give to multiple users of your MABRA systems:

SA - sets access; is used initially
LA - lists access
RA - removes access
CA - changes access
NEWPASS - lets the *CREATOR* enter his or her sign-on password to enable *UP* access for specified users. *NEWPASS* must be entered in System Mode.

When granting access to other users, the *CREATOR* should keep in mind that **critical commands** (i.e., commands that actually change the physical layout of the system) can be run only by the *CREATOR* when no update tasks (*UP* tasks) are pending and when there are no other users currently using the particular system. These critical commands are:

AF - add field
CF - change field
DF - delete field
RS - resize master file
RB - reblock master file
CS - copy a system

If you enter a critical command when updates are pending, a warning prints, telling you to run the updates before re-entering the critical command. If you enter a critical command and others are using the system, another warning prints, telling you the active user numbers. You can either wait awhile for the users to sign off, or locate the users and ask them to sign off until you process your critical command. No other users can enter this system until MABRA finishes processing the command.

1. Setting Access - *SA*

The *SA* command enables the *CREATOR* to give other users specific access to the current MABRA system. *SA* is appropriate to use as a way of setting initial access to your system.

When you enter *SA*, MABRA prompts with:

```
* SET ACCESS
* USER(S):
```

Enter a list of user numbers, separated by blanks (or enter *STOP* to return to the command prompt in Data Mode). Note that if any of these user numbers already has access to the system, his or her old access is **overwritten** by the new access you specify. Also note that since the *CREATOR*'s user number has complete access to the system, his or her access cannot be re-set. When you specify one or more user numbers, MABRA prompts with,

** USER(S) ARE* **user-number(s)**. *OK? (Y/N):*

where **user-number(s)** represents the list of authorized user numbers that you entered above. Enter *Y* to confirm that the list is correct; enter *N* to return to the *USER(S)* prompt; enter *STOP* to return to the command prompt in Data Mode.

If you enter *Y*, MABRA prompts with:

```
* COMMAND ACCESS
```

** COMMANDS:*

Enter the list of commands, separated by commas, that the specified users can have access to; enter *ALL* to specify all commands; enter *STOP* to return to the *USER(S)* prompt.

If you specify either a list of commands or *ALL*, MABRA prompts with:

** RECORD ACCESS*
** CONSTRAINTS:*

Enter a valid MABRA constraint to identify the group of records that the specified users can have access to (or enter *STOP* to return to the *USER(S)* prompt).

If you enter a valid constraint, MABRA responds with:

n *FOUND.*
** IS THIS CONSTRAINT OK? (Y/N):*

Enter *Y* if the number of records seems to be correct; enter *N* to ignore the constraint and return to the *CONSTRAINTS* prompt.

If you enter *Y*, MABRA prompts for specifications of field access with:

** FIELD ACCESS*

Recall that field access is broken down into *AR*, *CR*, and *LR* field access.

If the specified users have access to the *AR* command, MABRA issues the prompts:

> ** AR FIELD ACCESS*
> ** FIELDS:*
>
> Enter the fields that the specified users can have access to when using *AR*; enter a blank space and press carriage return to specify no fields; enter *ALL* to specify all fields; enter *STOP* to return to the *USER(S)* prompt. MABRA then returns with:
>
> ** MORE? (Y/N):*
>
> This prompt allows you to enter more fields if the input from the previous prompt was too long to fit on one line. Enter *Y* if more fields are to be entered; enter *N* if all necessary fields have been entered; enter *STOP* to return to the *USER(S)* prompt.

If the specified users have access to the *CR* or *CD* command, MABRA then prompts with:

> ** CR FIELD ACCESS*
> ** FIELDS:*
>
> Enter the fields that the specified users can have access to when using *CR* or *CD*; enter a blank space and press carriage return to specify no fields; enter *ALL* to specify all fields; enter *STOP* to return to the *USER(S)* prompt. If you enter field names, MABRA responds with *MORE? (Y/N)*, as described under *AR FIELD ACCESS*.

If the specified users have access to the *DR*, *CR*, *CD*, *BP*, *LR*, or *AN* commands, MABRA prompts with:

```
* LR FIELD ACCESS
* FIELDS:
```

Enter the fields that the specified users can have access to when using the *LR* command; enter a blank space and press carriage return to specify no fields; enter *ALL* to specify all fields; enter *STOP* to return to the *USER(S)* prompt. If you enter field names, MABRA responds with *MORE? (Y/N)*, as described under *AR FIELD ACCESS*.

After you set the various types of access that you wish to give specified users, MABRA prompts with:

```
* SET ACCESS? (Y/N):
```

Enter *Y* to set the specified access for the specified users; enter *N* to cancel the *SA* command and to return to the *USER(S)* prompt; enter *STOP* to cancel the *SA* command and to return to the command prompt in Data Mode.

If you enter *Y*, MABRA confirms that the access is set for the specified users by printing the message:

```
* ACCESS SET.
```

MABRA then returns to the *USER(S)* prompt.

2. Listing Access - *LA*

The *LA* command lists the types of access that a user has for the current MABRA system. If the *CREATOR* runs this command, he or she can list the status of every user who has access to the current system. When you enter *LA*, MABRA responds with:

```
* LIST ACCESS
* ENTER USER NUMBER(S):
```

Enter the user numbers, separated by blanks, of users whose access you wish to list; enter *ALL* to list the access types for all users; enter *STOP* to return to the command prompt in Data Mode. If you enter a list of user numbers or *ALL*, MABRA prints the following to prepare you for output:

```
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

Set up the paper for output, enter a blank space and press carriage return to print the access information at your terminal. After your output prints, again enter a blank space and press carriage return to get back to the command prompt in Data Mode.

3. Removing Access - *RA*

The *RA* command removes a specified user's access to the current MABRA system. When you enter *RA*, MABRA prompts with:

```
* REMOVE ACCESS
* USER(S):
```

Enter a list of user numbers, separated by blanks (or enter *STOP* to return to the command prompt in Data Mode). If you enter one or more user numbers, MABRA responds with,

** USER(S):* **user-number (s)** *OK? (Y/N):*

where **user-number (s)** represents the user number(s) you entered at the *USER(S)* prompt. Enter *Y* to confirm that these numbers are correct; enter *N* to cancel the user numbers and return to the *USER(S)* prompt; enter *STOP* to return to the command prompt in Data Mode.

If you enter *Y*, MABRA prints the message,

** ACCESS REMOVED.*

to confirm that the specified user(s) no longer have access to the current system. MABRA then returns to the *USER(S)* prompt.

Note that if you are currently using a MABRA system that you have been given access to, and the *CREATOR* presently removes your access to this system, a *FILE ACCESS ERROR* results when you try to run any of the commands.

4. Changing Access - *CA*

The *CA* command changes the access that specified users can have to the current system. *CA* is appropriate to use when you wish to change a portion of the access that you initially set with the *SA* command. You must be sure that your input to the *CA* command prompts is **complete** because *CA* does not add changes to the access already set by *SA*, but **overwrites** the access set by *SA*. When you enter *CA*, MABRA prompts with:

** CHANGE ACCESS*
** USER(S):*

Enter a list of user numbers, separated by blanks; enter *ALL* to change access for all users; enter *STOP* to return to the command prompt in Data Mode.

Note that since the *CREATOR*'s user number has complete access to the system, his or her access cannot be changed.

If you enter one or more user numbers, MABRA responds with:

** USER(S):* **user-number (s)** *OK? (Y/N):*

user-number (s) represents the user number(s) you entered at the *USER(S)* prompt. Enter *Y* to confirm that these numbers are correct; enter *N* to return to the *USER(S)* prompt; enter *STOP* to return to the command prompt in Data Mode.

If you enter *Y*, MABRA responds with:

** CHANGE: COMMAND, RECORD, OR FIELD ACCESS? (C/R/F):*

Enter *C* to change command access; enter *R* to change record access; enter *F* to change field access; enter *STOP* to return to the *USER(S)* prompt.

If you enter *C*, MABRA responds with:

```
* CHANGE COMMAND ACCESS
* COMMANDS:
```

Enter the commands, separated by commas, that you wish to give users access to; enter *ALL* to give access for all valid commands; enter *STOP* to return to the *CHANGE: COMMAND, RECORD, OR FIELD ACCESS...* prompt.

If you enter valid commands or *ALL*, MABRA accepts your input and returns with:

```
* CHANGE: COMMAND, RECORD, OR FIELD ACCESS? (C/R/F):
```

At this point, you can either change another level of access or enter *STOP*. If you enter *STOP*, MABRA responds with:

```
* CHANGE ACCESS? (Y/N):
```

thereby asking if you are finished with your changes and are ready to update the access.

If you enter *Y*, MABRA responds with,

```
* ACCESS CHANGED.
```

and returns to the *USER(S)* prompt.

If you enter *N* or *STOP*, MABRA ignores the previous input and returns to the *USER(S)* prompt.

If you enter *R*, MABRA responds with:

```
* CHANGE RECORD ACCESS
* CONSTRAINTS:
```

Enter a valid MABRA constraint to identify the records you wish to give specified users access to (or enter *STOP* to return to the *CHANGE: COMMAND, RECORD, OR FIELD ACCESS...* prompt).

If you enter a valid constraint, MABRA responds with:

```
n FOUND.
* IS THIS CONSTRAINT OK? (Y/N):
```

Enter *Y* if the number of records seems to be correct; enter *N* to cancel the constraint and to prompt for *CONSTRAINTS* again; enter *STOP* to ignore the constraint and to return to the *CHANGE: COMMAND, RECORD, OR FIELD ACCESS...* prompt.

If you enter *Y*, MABRA responds with:

```
* CHANGE: COMMAND, RECORD, OR FIELD ACCESS? (C/R/F):
```

If you enter *F*, MABRA responds with:

```
* CHANGE FIELD ACCESS
* CHANGE: ADD, CHANGE, OR LIST FIELD ACCESS? (A/C/L):
```

Enter either *A*, *C*, or *L* (or enter *STOP* to return to the *CHANGE: COMMAND, RECORD, OR FIELD ACCESS...* prompt.)

If you enter *A*, MABRA responds with:

```
* CHANGE AR FIELD ACCESS
* FIELDS:
```

Enter a list of field names, separated by commas, to indicate the fields that specified users can **add** values for; enter a blank space and a carriage return to bypass field input.

If you enter field names, MABRA responds with,

```
* MORE? (Y/N):
```

to prompt for more field names, if necessary. Enter *Y* if you wish to enter more field names; enter *N* to indicate that your field input is complete and to return to the *CHANGE: ADD, CHANGE, OR LIST FIELD ACCESS...* prompt; enter *STOP* to cancel the field name input and return to the *CHANGE: ADD, CHANGE, OR LIST FIELD ACCESS...* prompt.

If you enter *C* to the *CHANGE: ADD, CHANGE, OR LIST FIELD ACCESS? (A/C/L)* prompt, MABRA responds with:

```
* CHANGE CR FIELD ACCESS
* FIELDS:
```

Enter a list of field names, separated by commas, to indicate the fields that specified users can **change** values for; enter a blank space and a carriage return to bypass field input.

If you enter field names, MABRA responds with *MORE? (Y/N)*, as described under *CHANGE AR FIELD ACCESS*.

If you enter *L* to the *CHANGE: ADD, CHANGE, OR LIST FIELD ACCESS...* prompt, MABRA responds with:

```
* CHANGE LR FIELD ACCESS
* FIELDS:
```

Enter a list of field names, separated by commas, to indicate the fields that specified users can **list** values for; enter a blank space and a carriage return to bypass field input.

If you enter field names, MABRA responds with *MORE? (Y/N)*, as described under *CHANGE AR FIELD ACCESS*.

At this point, if you have finished making all your changes, enter *STOP*. MABRA responds with:

```
* CHANGE ACCESS? (Y/N):
```

As mentioned earlier, enter *Y* to process your changes; enter *N* or *STOP* to ignore the input and to return to the *USER(S)* prompt. If you enter *Y*, MABRA responds with,

```
* ACCESS CHANGED.
```

and returns to the *USER(S)* prompt.

5. Entering a New Sign-on Password - *NEWPASS*

The *NEWPASS* command stores the *CREATOR*'s new sign-on password—securely within a locked function—in the MABRA program. (See *SUBMITREQUEST* in the chapter, "File Documentation".) Using *NEWPASS* to maintain the correct password within MABRA is **only** necessary if the *CREATOR* gives *UP* access, with the *SA* or *CA* commands, to other users. *NEWPASS* applies to all of the *CREATOR*'s systems.

NEWPASS must be performed within System Mode, and only a MABRA system's *CREATOR* can enter the *NEWPASS* command. To enter *NEWPASS*, respond to the command prompt (:) as shown below:

```
* SYSTEM MODE
:NEWPASS
```

MABRA responds with,

```
* ENTER YOUR SIGNON PASSWORD: blot
```

where **blot** is a patch of overstruck characters. Enter your new sign-on password within the blotted patch (or enter *STOP* to return to the command prompt in System Mode). If you enter your password incorrectly, an error message results. MABRA can never learn what your password is, but only test the password for correctness. When you enter your password correctly, MABRA prints the message:

```
* PASSWORD UPDATED.
```

MABRA then returns to the command prompt in System Mode.

HOW TO SAVE MONEY

Using Absolute Record Numbers

Every record in a MABRA system has a unique number attached to it, called an **absolute record number**. This number represents the location of the record in the master file. By using an absolute record number to identify a record, you **eliminate the cost** of searching the system with various constraints.

You print a record's absolute record number—when listing a record with *LR*, *BP*, *CR*, or *DR*—by specifying a print field of delta (Δ), along with the other fields you wish to print. For example,

```
 CONSTRAINTS: SALARY>15000
    4 FOUND.
* PRINT FIELDS: Δ,SURNAME,INITIALS,SALARY
* SORT FIELDS: SURNAME
* ALIGN PAPER, TYPE SPACE, CR TO START.
  Δ SURNAME    INITIALS SALARY  Δ

 63 ACME       JB         16262
125 BLACK      AC         20151
  4 HUNT       HE         18562
 99 WOOLWORTH  WF         45112
```

lists the absolute record number and the values of the *SURNAME*, *INITIALS*, and *SALARY* fields for all records whose *SALARY* field is greater than 15000. In this case, MABRA sorts the information to print in alphabetical order by *SURNAME* so that, given a surname, you can quickly locate its absolute record number.

To locate records by their absolute record numbers, you enter a list of these numbers, separated by blanks, at the *CONSTRAINTS* prompt. For example,

```
* CONSTRAINTS: 4 63 27
    3 FOUND.
* PRINT FIELDS: SURNAME,INITIALS,SALARY,Δ
* SORT FIELDS: SALARY
* ALIGN PAPER, TYPE SPACE, CR TO START.
SURNAME    INITIALS SALARY   Δ

JONES       BR        14001   27
ACME        JB        16262   63
HUNT        HE        18562    4
```

lists the *SURNAME*, *INITIALS*, and *SALARY* fields of the records identified by absolute record numbers 4, 63, and 27.

Another way to get a listing of absolute record numbers is to batch print (*BP*) a portion, or all, of the data base by specifying a Δ among the desired print fields, and by then sorting the records along major fields (e.g., *SURNAME*, *INITIALS*). With this method, you use only one constraint to produce a reference listing that is sorted for easy identification of absolute record numbers.

Note that if you reference the absolute record number of a "deleted" (see *UP* and *RB*) record, or reference a record to which you do not have access in a multi-user system, MABRA ignores the reference to the absolute record number.

Note further that if you reblock the master file (*RB*) and choose to purge "deleted" records, you will thereby condense the master file and **change** its absolute record numbers. In other words, purging "deleted" records invalidates any previous listing of absolute record numbers from the master file.

Using *PREVIOUS* in a Constraint

If you wish to identify records within those found by the **previous** constraint, you can use the feature, *PREVIOUS*, in your new constraint. For example, the constraint,

```
PREVIOUS∧DEPT=SALES
```

identifies all records from the previous constraint whose *DEPT* fields have a value of *SALES*.

You can also use *PREVIOUS* if you want to change to a different command in Data Mode—say, from *LR* to *CR*—but still want to access the records from the previous command. When *CR* prompts for a constraint, you can use *PREVIOUS* as part of this constraint and thereby access the records found in *LR*'s constraint.

Inverting Fields

If you use certain fields regularly as search keys, these fields should be organized so that MABRA uses the least possible amount of computer time during a search. The method MABRA uses is to invert field values and store them in a file that optimizes searching. To invert a field, you must use the *CF* (change field) command, discussed in the chapter, "Data Mode".

You should avoid inverting fields that are not used regularly as search keys because you are storing them a second time for no reason. In such a case, update costs are higher because the field needs updating in two places.

Reblocking the Master File

The records of data for a system are stored in the master file. The blocking factor (number of records per file component) is set initially when MABRA creates a system. This value is based on a default, average record size that may not be suitable for your system. In such a case, reblocking the master file increases the efficiency of record searches and retrievals. To reblock the master file, you must use the *RB* command, discussed in the chapter, "Miscellaneous Commands".

Because *RB* is an expensive command to run if there are many records in the system, the best time to run *RB* is just after you define all your fields and before you add records to the system.

Stacking Input

This technique allows you to enter multiple commands and responses on the same line of input. You must use the diamond (◇) feature to **separate** your commands and responses. You can enter several commands together, as shown below,

```
* SYSTEM MODE
:ENTER◇1◇LR
```

where you enter Data Mode, choose system 1, and list its records.

You can stack your responses to commands which have multiple prompts, and thereby enter your responses while bypassing their prompts. For example, during the *AR* command, instead of responding to separate field prompts of *SURNAME*, *INITIALS*, and *SALARY*, you can enter:

```
AR◇CLOUTIER◇JP◇18000
```

With stacked input, you save money by reducing character output, computer processing time, and your own effort.

Using *CR* Versus *CD*

CR differs from *CD* in that *CR* changes **all** values of a particular field in a specified group of records to the **same, new** value, whereas *CD* changes **each** value of a particular field in a specified group of records to a **different, new** value.

Whether you should use *CR* or *CD* depends on the nature of the change and the number of searches (using constraints) you would perform to locate the records. For example, if you need to change a *SALARY* field in two or more personnel records, using *CD* with an inclusive constraint is cheaper (you search only once) than using *CR* and a constraint that must search each time for a single record.

Organizing Your Updates

When planning to run updates on your system, you should try to group updates together. For example, the cost of processing an update request to add **one** new record is almost the same as an update request to add **50** records. Therefore, you should try to enter as many records as you can in one *AR* update request. The same rule applies when using *CD*.

You should try also to avoid jumping indiscriminately from one command to another. Such jumping around increases the operating cost, considering the system overhead required by APL and MABRA when processing completely unrelated commands.

Setting Field Orders

The *FO* (field order) command optimizes the sequencing of field input during the *AR* (add record) command. You can set alternative field orders for entering field values, and then invoke the particular field order that is appropriate at the time of data entry. By using field ordering, you reduce the amount of computer time required to prompt for all field values and you eliminate unnecessary character output.

EXAMPLES SHOWING HOW TO USE 86 *MABRA*

1. Basic Commands

This section demonstrates how an ordinary user like myself can create and manipulate a MABRA system. To begin, I load the appropriate MABRA workspace and start the MABRA program. An asterisk (*) at the beginning of a MABRA statement indicates either a system response or a system prompt. I know that a MABRA statement is a prompt if it ends with a colon (:).

```
      )LOAD 86 MABRA
SAVED 17.53.43   06/07/79
      START
* MABRA. VERSION 2.
* SYSTEM MODE
```

The naked colon prompts for System Mode commands. I enter a command—in the following case, to create a MABRA system—immediately to the right of the colon.

```
:CREATE
* CREATE SYSTEM
* ENTER SYSTEM NUMBER (<100 AND >0): 37
* ENTER NAME FOR SYSTEM 37: EXAMPLE SYSTEM
* ENTER EXPECTED MAXIMUM NUMBER OF RECORDS: 100
* SYSTEM 37: EXAMPLE SYSTEM
* CREATE? (Y/N): Y
* SYSTEM CREATED.
```

While I'm in System Mode, I want to list the system numbers and corresponding names of all the systems on my user number.

```
:LIST
* LIST SYSTEM(S)
* ALIGN PAPER, TYPE SPACE, CR TO START.

MABRA SYSTEMS ON USER: 2345678  DATE: 17/10/79

NO.  NAME

37  EXAMPLE SYSTEM
```

Still within System Mode, I want to access the system that I just created.

```
:ENTER
* SYSTEM NUMBER: 37
* DATA MODE, SYSTEM IS 37: EXAMPLE SYSTEM
```

Now that I'm in Data Mode, I'm ready to add and define the fields that will structure the system: *SURNAME*, *INITIALS*, *DOBIRTH*, *SEX*, *SALARY*.

```
37:AF
* ADD FIELD
* ENTER FIELD NAME: SURNAME
* TYPE? (CHAR/INTE/REAL/BINA/DATE): CHAR
* NUMBER OF ELEMENTS IN FIELD: 15
* USE DEFAULT OUTPUT FORMAT? (Y/N): Y
* OK TO ADD? (Y/N): Y
* FIELD ADDED.
* ENTER FIELD NAME: INITIALS
* TYPE? (CHAR/INTE/REAL/BINA/DATE): CHAR
* NUMBER OF ELEMENTS IN FIELD: 3
* USE DEFAULT OUTPUT FORMAT? (Y/N): Y
* OK TO ADD? (Y/N): Y
* FIELD ADDED.
* ENTER FIELD NAME: DOBIRTH
* TYPE? (CHAR/INTE/REAL/BINA/DATE): DATE
* USE DEFAULT OUTPUT FORMAT? (Y/N): Y
* OK TO ADD? (Y/N): Y
* FIELD ADDED.
* ENTER FIELD NAME: SEX
* TYPE? (CHAR/INTE/REAL/BINA/DATE): CHAR
* NUMBER OF ELEMENTS IN FIELD: 1
* USE DEFAULT OUTPUT FORMAT? (Y/N): Y
* OK TO ADD? (Y/N): Y
* FIELD ADDED.
* ENTER FIELD NAME: SALARY
* TYPE? (CHAR/INTE/REAL/BINA/DATE): INTE
* NUMBER OF ELEMENTS IN FIELD: 1
* USE DEFAULT OUTPUT FORMAT? (Y/N): N
* ENTER ⎕FMT ARGUMENT: CI6
* OK TO ADD? (Y/N): Y
* FIELD ADDED.
* ENTER FIELD NAME: STOP
```

After I add the fields to the system, I want to list the fields and study their definitions.

```
37:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.

                  37: EXAMPLE SYSTEM      DATE: 17/10/79

NAME              TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

SURNAME           CHAR    15            ACL              15A1
INITIALS          CHAR     3            ACL              3A1
DOBIRTH           DATE     1            ACL              1G⎕ZZ/ZZ/Z9⎕
SEX               CHAR     1            ACL              1A1
SALARY            INT      1            ACL              CI6
```

I'm satisfied that my fields are correct, so now I'm ready to add records to the system. These records are not added directly to the system, but are compiled into an update request.

```
37:AR
* ADD RECORD

SURNAME  :PIKE
INITIALS :J
DOBIRTH  :131058
SEX      :M
SALARY   :18000

SURNAME  :SMITH
INITIALS :CC
DOBIRTH  :010452
SEX      :F
SALARY   :15600

SURNAME  :PATTISON
INITIALS :MJ
DOBIRTH  :190657
SEX      :F
SALARY   :14300

SURNAME  :LAWRASON
INITIALS :VL
DOBIRTH  :030450
SEX      :M
SALARY   :19400

SURNAME  :DOYLE
INITIALS :JF
DOBIRTH  :211256
SEX      :M
SALARY   :13500

SURNAME  :STOP
* UPDATE REQUEST NO. 1 FILED.
```

Now that the update request is filed, I can process it (thereby adding these records to the system) by initiating an update task. I choose an N-task because I want the update to run immediately, and I don't mind that it costs more than a B-task.

```
37:UP
* INITIATE UPDATE TASK
* 1 REQUEST PENDING UPDATE.
* RUN AS N-TASK OR B-TASK? (N/B): N
* N-TASK NO. 6577
```

To find out if my update N-task has completed, I take a new snapshot of my Data Mode session. The new shapshot further allows me to see the system changes made by the update.

```
37:NS
* UP N-TASK 6577 COMPLETED AT: 19:05:35 17/10/79
* SNAPSHOT TAKEN.
```

After taking a snapshot, I can search for and list the records that I added to the system. First, I enter a constraint that finds a particular value in the *SURNAME* field; then I enter the names of the fields I wish to print.

```
37:LR
* LIST RECORDS
* CONSTRAINTS: SURNAME=PIKE
    1 FOUND.
* PRINT FIELDS: SURNAME,INITIALS,SEX,SALARY
* SORT FIELDS:
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME         INITIALS SEX SALARY

PIKE            J        M   18,000
```

Now I want to search a field for a value that exceeds a certain amount. When I find the records that satisfy my constraint, I can print whatever fields I choose. Given that today's date is October 17, 1979, I also want to derive time periods based on a field that has date values. I can then sort the fields I wish to print, based on the increasing or decreasing values of a particular field.

```
* CONSTRAINTS: SALARY≥15000
    3 FOUND.
* PRINT FIELDS: SURNAME,SALARY,SEX,DOBIRTH,αDOBIRTH
* SORT FIELDS: -SALARY
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME         SALARY SEX  DOBIRTH αDOBIRTH

LAWRASON        19,400 M    3/04/50       29
PIKE            18,000 M   13/10/58       21
SMITH           15,600 F    1/04/52       27
```

Unique abbreviations can be used to identify any field name or its derivative, so I decide to use such abbreviations in the next constraint.

```
* CONSTRAINTS: SU=P∧SA>13600
    2 FOUND.
* PRINT FIELDS: SU,SA,D,SE,αD
* SORT FIELDS: SE,SA
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME         SALARY  DOBIRTH SEX αDOBIRTH

PATTISON        14,300 19/06/57 F         22
PIKE            18,000 13/10/58 M         21

* CONSTRAINTS: STOP
```

After listing the records, I find that one record (*PATTISON*) has errors in it. I change the record by correcting the erroneous field values. MABRA then files the field changes in an update request.

```
37:CR
* CHANGE RECORD(S)
* CONSTRAINTS: SURNAME=PATTISON
   1 FOUND.
* PRINT? (Y/N): Y
* PRINT FIELDS: ALL
* SORT FIELDS:
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME          INITIALS  DOBIRTH SEX SALARY

PATTISON         MJ       19/06/57 F   14,300

* ENTER CHANGES.
:SALARY←14700
:INITIALS←JB
:STOP
* LIST CHANGES? (Y/N): Y
SALARY          ←14700
INITIALS        ←JB
* FILE UPDATE REQUEST? (Y/N): Y
* UPDATE REQUEST NO. 2 FILED.
* CONSTRAINTS: STOP
```

After I change the record, it's appropriate for me to run another update request.

```
37:UP
* INITIATE UPDATE TASK
* 1 REQUEST PENDING UPDATE.
* RUN AS N-TASK OR B-TASK? (N/B): N
* N-TASK NO. 6615
```

By taking a new snapshot, I can tell whether the update has completed. If so, I get an updated version of Data Mode.

```
37:NS
* UP N-TASK 6615 COMPLETED AT: 19:09:52 17/10/79
* SNAPSHOT TAKEN.
```

Now I want to list all records, both to see if they are correct, and to derive the ages of all people in the system.

```
37:LR
* LIST RECORDS
* CONSTRAINTS: ALL
   5 FOUND.
* PRINT FIELDS: ALL,αDOB
* SORT FIELDS: SURNAME
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
SURNAME          INITIALS  DOBIRTH SEX SALARY αDOBIRTH

DOYLE            JF        21/12/56 M   13,500        22
LAWRASON         VL         3/04/50 M   19,400        29
PATTISON         JB        19/06/57 F   14,700        22
PIKE             J         13/10/58 M   18,000        21
SMITH            CC         1/04/52 F   15,600        27

* CONSTRAINTS: STOP
```

At this point, I want to delete a record, but to print the record beforehand. Whenever records are deleted, MABRA generates an update request.

```
37:DR
* DELETE RECORD(S)
* CONSTRAINTS: SU=LAWRASON∧I=VL
    1 FOUND.
* PRINT RECORDS TO BE DELETED? (Y/N): Y
* PRINT FIELDS: ALL
* SORT FIELDS:
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME          INITIALS  DOBIRTH SEX SALARY

LAWRASON         VL         3/04/50 M   19,400

* FILE UPDATE REQUEST? (Y/N): Y
* UPDATE REQUEST NO. 3 FILED.
* CONSTRAINTS: STOP
```

When I run the update request I can immediately specify the type of task I wish to run by using the diamond (◇) feature to answer the *RUN AS AN N-TASK OR B-TASK? (N/B)* prompt before it occurs.

```
37:UP◇N
* 1 REQUEST PENDING UPDATE.
* N-TASK NO. 6628
```

Taking a new snapshot shows that the update has completed and returns an updated version of Data Mode.

```
37:NS
* UP N-TASK 6628 COMPLETED AT: 19:11:31 17/10/79
* SNAPSHOT TAKEN.
```

Now I want to add more records to the system. If I don't have a value for a field, I can go to another field by using the "jump to" (→) feature. Then, once I become familiar with the input pattern for field values, I can use the ◇ feature to answer field prompts before they occur.

```
37:AR
* ADD RECORD

SURNAME  :POLONIO
INITIALS :MM
DOBIRTH  :130358
SEX      :M
SALARY   :12000

SURNAME  :MAY
INITIALS :BJ
DOBIRTH  :100154
SEX      :F
SALARY   :20000

SURNAME  :MCDUFF
INITIALS :→SALARY
SALARY   :14234

SURNAME  :SMITH
INITIALS :MR
DOBIRTH  :121245
SEX      :→SALARY
SALARY   :18777

SURNAME  :DOE◇J◇010162◇M◇17600

SURNAME  :BLOGGS◇FJ◇010150◇F◇9000

SURNAME  :STOP
* UPDATE REQUEST NO. 4 FILED.
```

I decide to update the system with a *UP* N-task. After that, I'll update my version of Data Mode with a new shapshot.

```
37:UP◇N
* 1 REQUEST PENDING UPDATE.
* N-TASK NO. 6654
37:NS
* UP N-TASK 6654 COMPLETED AT: 19:14:57 17/10/79
* SNAPSHOT TAKEN.
```

Now I want to list all the records on the system. I decide to sort the field values both by *SEX* (ladies first) and by *SURNAME* (in reverse alphabetical order). Note that there are three sorted levels in the output: the first two records are exceptional because one of their sort fields (*SEX*) has no value; the next four records are ordered by their *SEX* values being female; the remaining four records are ordered as male. Note further that within all three levels of output, sorting also occurs along the second sort field indicated—reverse alphabetical order by *SURNAME*.

```
37:LR
* LIST RECORDS
* CONSTRAINTS: ALL
  10 FOUND.
* PRINT FIELDS: ALL
* SORT FIELDS: SEX,-SU
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME          INITIALS  DOBIRTH SEX SALARY

SMITH            MR       12/12/45      18,777
MCDUFF                           0      14,234
SMITH            CC        1/04/52 F    15,600
PATTISON         JB       19/06/57 F    14,700
MAY              BJ       10/01/54 F    20,000
BLOGGS           FJ        1/01/50 F     9,000
POLONIO          MM       13/03/58 M    12,000
PIKE             J        13/10/58 M    18,000
DOYLE            JF       21/12/56 M    13,500
DOE              J         1/01/62 M    17,600

* CONSTRAINTS: STOP
```

Now I'd like to add a new field (*DINCREASE*) that contains the date of an increase in salary.

```
37:AF
* ADD FIELD
* ENTER FIELD NAME: DINCREASE
* TYPE? (CHAR/INTE/REAL/BINA/DATE): DATE
* USE DEFAULT OUTPUT FORMAT? (Y/N): Y
* OK TO ADD? (Y/N): Y
* FIELD ADDED.
* ENTER FIELD NAME: STOP
```

I want to list the field definitions to see if I made the field addition correctly.

```
37:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.

                  37: EXAMPLE SYSTEM  DATE: 17/10/79

NAME              TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

SURNAME           CHAR    15            ACL              15A1
INITIALS          CHAR     3            ACL              3A1
DOBIRTH           DATE     1            ACL              1G⎕ZZ/ZZ/Z9⎕
SEX               CHAR     1            ACL              1A1
SALARY            INT      1            ACL              CI6
DINCREASE         DATE     1            ACL              1G⎕ZZ/ZZ/Z9⎕
```

Now I want to give all females in the system a well-deserved raise. To accomplish this, I have to change the *SALARY* and *DINCREASE* fields. To help me change each record correctly, I decide to use the *SURNAME* and *INITIALS* fields as print fields for identifying each record I want to change. I further decide to sort the print fields so that I am prompted with the *SURNAME* and *INITIALS* fields in alphabetical order.

```
37:CD
* CHANGE DIFFERENT VALUES
* CHANGE FIELDS: SALARY,DINC
* PRINT FIELDS: SURNAME,INITIALS
* SORT FIELDS: SU,INI
* CONSTRAINTS: SEX=F
   4 FOUND.

* BLOGGS          ;FJ
  SALARY     :10000
  DINCREASE :061079

* MAY             ;BJ
  SALARY     :22000
  DINCREASE :061079

* PATTISON        ;JB
  SALARY     :17000
  DINCREASE :071079

* SMITH           ;CC
  SALARY     :24000
  DINCREASE :131079
* UPDATE REQUEST NO. 5 FILED.
* CONSTRAINTS: STOP
* CHANGE FIELDS: STOP
```

Now I process the update task generated by *CD*, take a new snapshot of the records in Data Mode, and list all the records in the system, printing and sorting by certain fields.

```
37:UP◇N
* 1 REQUEST PENDING UPDATE.
* N-TASK NO. 6676
37:NS
* UP N-TASK 6676 COMPLETED AT: 19:17:34 17/10/79
* SNAPSHOT TAKEN
37:LR
* LIST RECORDS
* CONSTRAINTS: ALL
  10 FOUND.
* PRINT FIELDS: SUR,INI,αDOB,SAL,DINC
* SORT FIELDS: -DOB
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
SURNAME         INITIALS αDOBIRTH SALARY DINCREASE

DOE             J              17 17,600        0
PIKE            J              21 18,000        0
POLONIO         MM             21 12,000        0
PATTISON        JB             22 17,000  7/10/79
DOYLE           JF             22 13,500        0
MAY             BJ             25 22,000  6/10/79
SMITH           CC             27 24,000 13/10/79
BLOGGS          FJ             29 10,000  6/10/79
SMITH           MR             33 18,777        0
MCDUFF                            14,234        0

* CONSTRAINTS: STOP
```

I decide to change the *SALARY* field's output format so that it contains a floating dollar sign ($).

```
37:CF
* CHANGE FIELD
* FIELD NAME: SALARY
* CHANGE FIELD=SALARY
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT: ⎕FMT
* NEW ⎕FMT ARGUMENT: P⎕$⎕CI8
* FORMAT CHANGED.
* CHANGE FIELD=SALARY
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT: STOP
* FIELD NAME: STOP
```

Now I want to list all records having *SALARY* field values greater than a certain amount. In the process, I'll display the *SALARY* output format that I just changed.

```
37 LR
* LIST RECORDS
* CONSTRAINTS: SALARY>20000
   2 FOUND.
* PRINT FIELDS: SUR,SAL
* SORT FIELDS:
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME           SALARY

SMITH            $24,000
MAY              $22,000

* CONSTRAINTS: STOP
```

Next, I decide to change the name of a field from *SURNAME* to *LASTNAME*.

```
37:CF
* CHANGE FIELD
* FIELD NAME: SURNAME
* CHANGE FIELD=SURNAME
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT: NAME
* ENTER NEW FIELD NAME: LASTNAME
* OK TO CHANGE NAME FROM 'SURNAME' TO 'LASTNAME'? (Y/N): Y
* FIELD NAME CHANGED FROM 'SURNAME' TO 'LASTNAME.'
* CHANGE FIELD=LASTNAME
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT: STOP
* FIELD NAME: STOP
```

Now I want to list all records whose *SURNAME* fields equal *PIKE*. I am told that *SURNAME* is an invalid field name (since I changed it above) and then realize that I want to search the *LASTNAME* field instead. After specifying *PRINT FIELDS*, I enter three diamonds in order to answer (bypass) three intermediate prompts before they occur.

```
37:LR
* LIST RECORDS
* CONSTRAINTS: SURNAME=PIKE
? IDENTIFIER UNDEFINED
SURNAME=PIKE
^
* CONSTRAINTS: LASTNAME=PIKE
    1 FOUND.
* PRINT FIELDS: ALL◇ ◇ ◇
LASTNAME        INITIALS  DOBIRTH SEX   SALARY DINCREASE

PIKE            J         13/10/58 M    $18,000        0
* CONSTRAINTS: STOP
```

I decide to delete the *SEX* field, seeing that the system requires no distinction between male and female.

```
37:DF
* DELETE FIELD(S)
* FIELDS TO BE DELETED: SEX
* DELETE SEX? (Y/N): Y
* DELETED.
* FIELDS TO BE DELETED: STOP
```

Now that my field changes are complete, I want a new listing of the field definitions.

```
37:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
                   37: EXAMPLE SYSTEM      DATE: 17/10/79

NAME              TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

LASTNAME          CHAR    15            ACL              15A1
INITIALS          CHAR     3            ACL              3A1
DOBIRTH           DATE     1            ACL              1G⎕ZZ/ZZ/Z9⎕
SALARY            INT      1            ACL              P⎕$⎕CI8
DINCREASE         DATE     1            ACL              1G⎕ZZ/ZZ/Z9⎕
```

My work is now complete, so I exit MABRA by entering *STOP* after the command prompt (37:) in Data Mode, and again, after the command prompt (:) in System Mode.

```
37:STOP
* SYSTEM MODE
:STOP
```

2. Performing Crosstabulations with *AN*

This section shows how to perform various crosstabulation analyses by using the *AN* command. First, I load the appropriate MABRA workspace and start the MABRA program. I enter into Data Mode from System Mode. I then request MABRA system number 1, *ACME COMPUTER SERVICES.*

```
      )LOAD 86 MABRA
SAVED 17.53.43  06/07/79
      START
* MABRA. VERSION 2.
* SYSTEM MODE
:ENTER
* SYSTEM NUMBER: 1
* DATA MODE, SYSTEM IS 01: ACME COMPUTER SERVICES
```

Now I list the field definitions to see the structure of the fields.

```
01:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.

              01: ACME COMPUTER SERVICES      DATE: 18/10/79

NAME              TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

SURNAME           CHAR    10     Y      ACL              10A1
DOBIRTH           DATE     1     Y      ACL              1G⎕ZZ/ZZ/Z9⎕
SEX               CHAR     1     Y      ACL              1A1
DOJOIN            DATE     1            ACL              1G⎕ZZ/ZZ/Z9⎕
SALARY            REAL     1            ACL              CI7
LOCATION          CHAR     4     Y      ACL        25    4A1
```

Because I want a printed reference for checking certain analyses, I decide to list all the records in the system. I want to print all the fields in each record, as well as to print the codes for the *LOCATION* field. I want to sort the records in alphabetical order by *SURNAME.*

```
01:LR
* LIST RECORDS
* CONSTRAINTS: ALL
   52 FOUND.
* PRINT FIELDS: ALL,(LOC)
* SORT FIELDS: SURNAME
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
SURNAME      DOBIRTH SEX   DOJOIN  SALARY LOCATION (LOCATION)

ACME        16/10/21 M   11/05/39 100,000 0204     STIRLING, SCOTLAND
ALLEN        8/05/28 F   15/12/42  11,921 0102     BIRMINGHAM, ENGLAND
BROOKES     27/08/16 M   19/09/35  16,618 0403     ARMAGH, N IRELAND
BROWN        1/03/20 F    6/06/35   7,796 0104     LIVERPOOL, ENGLAND
BRUFFELL     8/08/18 M    1/06/58   7,447 0401     BELFAST, N IRELAND
BULLUCKS     4/07/26 M   11/10/43  12,999 0204     STIRLING, SCOTLAND
CASH        17/06/43 M   20/01/77  14,910 0402     LONDONDERRY, N IRELAND
CHAUHAN     12/08/56 M    1/08/78   7,500 0110     BOURNEMOUTH, ENGLAND
CLARKE      23/02/55 M   20/10/71  11,866 0202     GLASGOW, SCOTLAND
COOPER      18/06/29 F   20/08/49  14,913 0101     LONDON, ENGLAND
EVANS       11/10/29 M   30/08/71  35,000 0203     ABERDEEN, SCOTLAND
FELL        16/08/23 M    1/08/37  14,613 0101     LONDON, ENGLAND
FLANAGAN    30/08/32 F    3/01/49  14,547 0404     BALLYMORA, N IRELAND
FORD        29/06/28 M    7/04/44  14,965 0202     GLASGOW, SCOTLAND
FRANKLIN    26/02/18 F   28/01/36  13,433 0403     ARMAGH, N IRELAND
GARLICK     25/12/13 M   22/04/30  14,737 0503     WATERFORD, EIRE
HALL        21/04/15 M    4/10/29   7,606 0404     BALLYMORA, N IRELAND
HANNEY      21/10/19 M    1/10/50  13,843 0404     BALLYMORA, N IRELAND
HORTON       8/10/46 M   29/09/63  12,856 0403     ARMAGH, N IRELAND
HUTCHINSON  15/05/51 M   13/09/69  13,480 0101     LONDON, ENGLAND
JONES       28/08/39 M    3/01/66   9,663 0503     WATERFORD, EIRE
KIDSON      18/06/28 M    3/11/44  13,172 0202     GLASGOW, SCOTLAND
KILMISTER   11/01/22 F    7/01/39   8,453 0103     MANCHESTER, ENGLAND
LAW         20/12/39 M    2/08/59  17,960 0504     LIMERICK, EIRE
LEE         16/02/32 M   16/01/68   7,840 0203     ABERDEEN, SCOTLAND
LINDER      14/04/52 M   27/11/78  14,456 0404     BALLYMORA, N IRELAND
MARCHANT    13/06/27 M    4/03/57  14,104 0401     BELFAST, N IRELAND
METTRICK    22/01/27 M   23/12/54  12,647 0203     ABERDEEN, SCOTLAND
NOBLE       19/05/37 M    1/02/63  14,767 0403     ARMAGH, N IRELAND
NORLEDGE    14/01/44 M   18/02/65   9,289 0101     LONDON, ENGLAND
PHILLIPS    30/05/35 M    1/02/69  16,117 0110     BOURNEMOUTH, ENGLAND
PINKERTON   15/06/35 M    8/07/62  14,283 0402     LONDONDERRY, N IRELAND
PRICE        6/10/25 M    7/11/53  17,245 0110     BOURNEMOUTH, ENGLAND
ROBERTS      6/10/23 M    1/01/40  14,225 0502     CORK, EIRE
ROBERTSON    8/09/48 F   21/02/77  15,100 0101     LONDON, ENGLAND
RODGER      14/05/44 F   29/12/69  10,859 0403     ARMAGH, N IRELAND
RYAN        10/04/47 M   17/02/70  13,145 0503     WATERFORD, EIRE
SAUVARY     22/07/24 F    2/09/74  15,488 0201     EDINBURGH, SCOTLAND
SAVAGE       1/10/39 M    7/10/69  15,004 0504     LIMERICK, EIRE
SCHILLER    27/01/50 M    7/09/66  18,166 0503     WATERFORD, EIRE
SMITH        1/09/21 M   27/01/58  13,843 0102     BIRMINGHAM, ENGLAND
SMITH        2/03/44 M   10/07/61  42,500 0501     DUBLIN, EIRE
TILLION     27/12/30 M   22/02/67  15,149 0404     BALLYMORA, N IRELAND
TROOP       21/11/47 M    3/01/72  17,723 0110     BOURNEMOUTH, ENGLAND
WALKER      12/03/30 M   11/09/54  13,098 0204     STIRLING, SCOTLAND
WALKER      31/01/35 F   18/02/66  16,981 0102     BIRMINGHAM, ENGLAND
WALL        18/03/36 M   11/09/51  17,517 0402     LONDONDERRY, N IRELAND
WEBB         2/07/37 M   25/07/67  13,062 0502     CORK, EIRE
WEST        19/04/31 M   13/04/69  10,186 0403     ARMAGH, N IRELAND
WILLIAMS     4/06/50 M   23/02/72  14,737 0203     ABERDEEN, SCOTLAND
WILLIAMS    20/10/37 M   15/03/58  18,103 0403     ARMAGH, N IRELAND
WOOD        26/11/44 M   17/10/77  14,187 0404     BALLYMORA, N IRELAND
```

```
* CONSTRAINTS: STOP
```

Because some analyses will deal with values in the *LOCATION* field, I decide to print the code-description table for this field, sorted in ascending order by code.

```
01:LC
* LIST CODES/DESCS.
* FOR WHICH FIELD: LOCATION
* SORT BY CODE OR DESC. (C/D): C
* ALIGN PAPER, TYPE SPACE, CR TO START.

* CURRENT NUMBER OF CODES IS 16.
* SUGGESTED MAXIMUM NUMBER OF CODES IS 77.

0101  LONDON, ENGLAND
0102  BIRMINGHAM, ENGLAND
0103  MANCHESTER, ENGLAND
0104  LIVERPOOL, ENGLAND
0201  EDINBURGH, SCOTLAND
0202  GLASGOW, SCOTLAND
0203  ABERDEEN, SCOTLAND
0204  STIRLING, SCOTLAND
0401  BELFAST, N IRELAND
0402  LONDONDERRY, N IRELAND
0403  ARMAGH, N IRELAND
0404  BALLYMORA, N IRELAND
0501  DUBLIN, EIRE
0502  CORK, EIRE
0503  WATERFORD, EIRE
0504  LIMERICK, EIRE

* FOR WHICH FIELD: STOP
```

The first crosstabulation analysis that I do involves counting the frequency of the values (*M* and *F*) in the *SEX* field. Because there are only two values in *SEX*, they don't need separating into ranges. The result of the analysis shows the breakdown of the values in *SEX* according to the total number of records analyzed.

```
01:AN
* CROSSTABULATION ANALYSIS
* OPTION? (F/T/A/FT/FA/FTA/TA): F
* CONSTRAINTS: ALL
  52 FOUND.
* FIELD NAME LIST: SEX
* USE RANGES? (Y/N): N
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
                        01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: ALL, ATTRIBUTES: SEX.

SEX=F :      10
   =M :      42
TOTAL :      52
```

In my next analysis, I want to count the *SALARY* values that range, in increments of 5000, between 10000 and 30000, for all males.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): F
* CONSTRAINTS: SEX=M
   42 FOUND.
* FIELD NAME LIST: SALARY
* USE RANGES? (Y/N): Y
* FIELD NAMES ARE:
  1. SALARY
* ENTER FIELD NAME'S NUMBER: 1
* SALARY

:10000
:15000
:20000
:25000
:30000
:STOP
* ENTER FIELD NAME'S NUMBER: STOP
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.

                        01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: SEX=M, ATTRIBUTES: SALARY.

         SALARY<10000    :       6
10000  ≤SALARY<15000    :      23
15000  ≤SALARY<20000    :      10
20000  ≤SALARY<25000    :
25000  ≤SALARY<30000    :
30000  ≤SALARY          :       3
                 TOTAL :      42
```

In the next analysis, I want to count the frequencies of first letters in the *SURNAME* field for all records whose *SALARY* values are greater than 15000.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): F
* CONSTRAINTS: SALARY>15000
   16 FOUND.
* FIELD NAME LIST: SURNAME[1]
* USE RANGES? (Y/N): N
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
                  01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: SALARY>15000, ATTRIBUTES: SURNAME[1].

SURNAME[1]=A :       1
          =B :       1
          =E :       1
          =L :       1
          =P :       2
          =R :       1
          =S :       4
          =T :       2
          =W :       3
       TOTAL :      16
```

Now I want to count all the records whose *DOBIRTH* (date of birth) fields yield ages: less than 25 years, between 25 and 65 years (in five-year increments), and greater than 65 years. I can enter all α*DOBIRTH* range points at the first prompt by using the ◇ feature.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): F
* CONSTRAINTS: ALL
  52 FOUND.
* FIELD NAME LIST: αDOBIRTH
* USE RANGES? (Y/N): Y
* FIELD NAMES ARE:
  1. αDOBIRTH
* ENTER FIELD NAME'S NUMBER: 1
* αDOBIRTH

:25◇30◇35◇40◇45◇50◇55◇60◇65◇STOP
* ENTER FIELD NAME'S NUMBER: STOP
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.

                  01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: ALL, ATTRIBUTES: αDOBIRTH.

    αDOBIRTH< 25 :       2
 25≤αDOBIRTH< 30 :       4
 30≤αDOBIRTH< 35 :       5
 35≤αDOBIRTH< 40 :       5
 40≤αDOBIRTH< 45 :       9
 45≤αDOBIRTH< 50 :       5
 50≤αDOBIRTH< 55 :       9
 55≤αDOBIRTH< 60 :       8
 60≤αDOBIRTH< 65 :       4
 65≤αDOBIRTH     :       1
           TOTAL :      52
```

Next, I want to count all records to show the distribution of *SEX* by *LOCATION*. The result is a table of values based on the implicit constraint, *LOCATION* ∧ *SEX*. The *LOCATION* codes are represented along the left of the table. Abbreviated values (*C*1 and *C*2) for *SEX* are represented along the top of the table. A legend beneath the table converts the abbreviated values in the *SEX* constraint into their defined values (*M* and *F*). Such abbreviations are intended to help format output when field values are greater than a couple of elements.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): F
* CONSTRAINTS: ALL
   52 FOUND.
* FIELD NAME LIST: LOC,SEX
* USE RANGES? (Y/N): N
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.

                    01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: ALL, ATTRIBUTES: LOC,SEX.

            C1   C2  TOTAL
        +-----------------
LOC=0101|    2    3      5
   =0102|    2    1      3
   =0103|    1           1
   =0104|    1           1
   =0110|         4      4
   =0201|    1           1
   =0202|         3      3
   =0203|         4      4
   =0204|         3      3
   =0401|         2      2
   =0402|         3      3
   =0403|    2    5      7
   =0404|    1    5      6
   =0501|         1      1
   =0502|         2      2
   =0503|         4      4
   =0504|         2      2
   TOTAL|   10   42     52

  C1 : SEX=F
  C2 :    =M
```

In the next case, I want to analyze all records for the relationship between age (*αDOBIRTH*) and length of service (*αDOJOIN*).

```
* OPTION? (F/T/A/FT/FA/FTA/TA): F
* CONSTRAINTS: ALL
   52 FOUND.
* FIELD NAME LIST: αDOBIRTH,αDOJOIN
* USE RANGES? (Y/N): Y
* FIELD NAMES ARE:
  1. αDOBIRTH
  2. αDOJOIN
* ENTER FIELD NAME'S NUMBER: 1
* αDOBIRTH
```

```
:20◇30◇40◇50◇60
:STOP
* ENTER FIELD NAME'S NUMBER: 2
* αDOJOIN

:2◇4◇6◇8◇10◇12◇15
:STOP
* ENTER FIELD NAME'S NUMBER: STOP
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.

                      01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: ALL, ATTRIBUTES: αDOBIRTH,αDOJOIN.

                      C1   C2   C3   C4   C5   C6   C7   C8  TOTAL
                +------------------------------------------------
    αDOBIRTH< 20|
 20≤αDOBIRTH< 30|    2              2         1    1           6
 30≤αDOBIRTH< 40|         3         1    2         1    3     10
 40≤αDOBIRTH< 50|                             4    4    6     14
 50≤αDOBIRTH< 60|              1         1             15     17
 60≤αDOBIRTH    |                                       5      5
           TOTAL|    2    3    1    3    3    5    6   29     52


  C1 :      αDOJOIN<  2
  C2 :    2≤αDOJOIN<  4
  C3 :    4≤αDOJOIN<  6
  C4 :    6≤αDOJOIN<  8
  C5 :    8≤αDOJOIN< 10
  C6 :   10≤αDOJOIN< 12
  C7 :   12≤αDOJOIN< 15
  C8 :   15≤αDOJOIN
```

In the next analysis, I want to show the relationships among *SEX* by country (*LOCATION*[1 2]) by *SALARY* ranges.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): F
* CONSTRAINTS: ALL
   52 FOUND.
* FIELD NAME LIST: SEX,LOC[1 2],SALARY
* USE RANGES? (Y/N): Y
* FIELD NAMES ARE:
   1. SEX
   2. LOC[1 2]
   3. SALARY
* ENTER FIELD NAME'S NUMBER: 3
* SALARY
```

```
:10000◇12222◇15555◇17777◇20000◇30000
:STOP
* ENTER FIELD NAME'S NUMBER: STOP
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.

                   01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: ALL, ATTRIBUTES: SEX,LOC[1 2],SALARY.


SEX=F
              C1   C2   C3   C4   C5   C6   C7  TOTAL
           +----------------------------------------
LOC[1 2]=01|   2    1    1    2                    6
       =02|                  1                    1
       =04|        1    2                         3
       =05|
      TOTAL|   2    2    3    3                   10

----------------------------------------------------


SEX=M

              C1   C2   C3   C4   C5   C6   C7  TOTAL
           +----------------------------------------
LOC[1 2]=01|   2         3    2    1               8
       =02|   1    1    6                   2    10
       =04|   2    1    8    2    2              15
       =05|   1         4    1    2         1     9
      TOTAL|   6    2   21    5    5         3    42


  C1 :        SALARY  <10000
  C2 : 10000 ≤SALARY  <12222
  C3 : 12222 ≤SALARY  <15555
  C4 : 15555 ≤SALARY  <17777
  C5 : 17777 ≤SALARY  <20000
  C6 : 20000 ≤SALARY  <30000
  C7 : 30000 ≤SALARY
```

Now I want to analyze all records to total the *SALARY* fields by *LOCATION*.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): T
* CONSTRAINTS: ALL
   52 FOUND.
* FIELD NAME LIST: LOCATION
* USE RANGES? (Y/N): N
* FIELD TO BE TOTALLED: SALARY
* ROUND TO HOW MANY DECIMAL PLACES? 0
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.

                       01: ACME COMPUTER SERVICES
* TOTALS. CONSTRAINTS: ALL, ATTRIBUTES: LOCATION, TOTALLING  SALARY

LOCATION=0101 :      67395
        =0102 :      42745
        =0103 :       8453
        =0104 :       7796
        =0110 :      58585
        =0201 :      15488
        =0202 :      40003
        =0203 :      70224
        =0204 :     126097
        =0401 :      21551
        =0402 :      46710
        =0403 :      96822
        =0404 :      79788
        =0501 :      42500
        =0502 :      27287
        =0503 :      55711
        =0504 :      32964
        TOTAL :     840119
```

Below, I want to analyze all records in order to average the *SALARY* fields for each country (*LOCATION*[1 2]) by *SEX*. Notice that the column under the *TOTALS* heading represents **average totals**.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): A
* CONSTRAINTS: ALL
   52 FOUND.
* FIELD NAME LIST: LOCATION[1 2],SEX
* USE RANGES? (Y/N): N
* FIELD TO BE AVERAGED: SALARY
* ROUND TO HOW MANY DECIMAL PLACES? 0
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
                    01: ACME COMPUTER SERVICES
* AVERAGES. CONSTRAINTS: ALL, ATTRIBUTES: LOCATION[1 2],SEX, AVERAGING  SALARY
                           C1     C2      TOTAL
                 +------------------------
LOCATION[1 2]=01|  12527  13726    13212
             =02|  15488  23632    22892
             =04|  12946  13735    13604
             =05|         17607    17607
           TOTAL|  12949  16920    16156

 C1 : SEX=F
 C2 :    =M
```

In this analysis, I want to examine all records for their total values by country (*LOCATION*[1 2]) and their average values for age (α*DOBIRTH*) by country.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): FA
* CONSTRAINTS: ALL
  52 FOUND.
* FIELD NAME LIST: LOCATION[1 2]
* USE RANGES? (Y/N): N
* FIELD TO BE AVERAGED: αDOBIRTH
* DO ANALYSIS? (Y/N): Y
* ALIGN PAPER, TYPE SPACE, CR TO START.

                    01: ACME COMPUTER SERVICES
* FREQUENCY ANALYSIS. CONSTRAINTS: ALL, ATTRIBUTES: LOCATION[1 2].

LOCATION[1 2]=01 :     14
             =02 :     11
             =04 :     18
             =05 :      9
           TOTAL :     52

                    01: ACME COMPUTER SERVICES
* AVERAGES. CONSTRAINTS: ALL, ATTRIBUTES: LOCATION[1 2], AVERAGING αDOBIRTH

LOCATION[1 2]=01 :    45
             =02 :    48
             =04 :    47
             =05 :    42
           TOTAL :    46
```

I have finished making my crosstabulation analyses and now wish to leave the MABRA system.

```
* OPTION? (F/T/A/FT/FA/FTA/TA): QUIT
```

3. Coding Fields with *CF*

This section shows how to code a field in a MABRA system. To begin the session, I load the appropriate MABRA workspace and start the MABRA program. I enter Data Mode and request MABRA system number 2, *LOCATION FINDER*.

```
      )LOAD 86 MABRA
SAVED 17.53.43  06/07/79
      START
* MABRA. VERSION 2.
* SYSTEM MODE
:ENTER
* SYSTEM NUMBER: 2
* DATA MODE, SYSTEM IS 02: LOCATION FINDER
```

Below, I list the field definitions to see the structure of the fields. The *DESC-WID* (width of the code's description) column is empty, indicating there are no coded fields in the system.

```
02:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.

                02: LOCATION FINDER     DATE: 22/10/79

NAME            TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

MAILBOX         CHAR    5             ACL              5A1
LOCATION        INT     1             ACL              ZI4
```

Now I list all the records in the system. I want to print all the fields and sort them in alphabetical order by *MAILBOX*. *MAILBOX* contains a person's unique address; *LOCATION* contains a numeric value that indicates a physical location.

```
02:LR
* LIST RECORDS
* CONSTRAINTS: ALL
  14 FOUND.
* PRINT FIELDS: ALL
* SORT FIELDS: MAILBOX
* ALIGN PAPER, TYPE SPACE, CR TO START.
```

```
MAILBOX LOCATION

BJP        0303
DCO        0303
DINO       0302
DPM        0101
GHR        0501
HCA        1201
HHY        0101
JMC        0501
JMS        0101
KMC        0301
NICKT      0304
RAJ        0301
RSG        0201
SJT        1101

* CONSTRAINTS: STOP
```

Now that I have listed the system's fields and records and am familiar with them, I am ready to change the field definition of *LOCATION* so that it is coded. At the same time, MABRA sets up an empty code-description table. In *LOCATION*'s case, I want the codes' descriptions to have a width of 20 characters.

```
02:CF
* CHANGE FIELD
* FIELD NAME: LOCATION
* CHANGE FIELD=LOCATION
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT: CODE
* FIELD IS NOT CODED.
* CODE IT? (Y/N): Y
* CODE'S DESCRIPTION WIDTH: 20
* SUGGESTED MAXIMUM NUMBER OF CODES IS 100.
* OK? (Y/N): Y
* CODED.
* CHANGE FIELD=LOCATION
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT: STOP
* FIELD NAME: STOP
```

Now I want to list the field definitions again. The *DESC-WID* column contains the value 20 for *LOCATION*—confirming the coded status.

```
02:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.

                02: LOCATION FINDER     DATE: 22/10/79

NAME       TYPE WIDTH  INVERTED ACCESS DESC-WID  ⎕FMT

MAILBOX    CHAR     5           ACL                5A1
LOCATION   INT      1           ACL       20       ZI4
```

At this point I'm ready to add descriptions to the code-description table. The system prompts for a code value and then for its description. All codes and descriptions that I enter are updated to the code-description table.

```
02:AC
* ADD CODES/DESCS.
* FOR WHICH FIELD: LOCATION
* CURRENT NUMBER OF CODES IS 0.
* MAXIMUM NUMBER OF CODES IS 100.
* CODE: 301
* DESC.: LONDON, ENGLAND
* CODE: 302
* DESC.: GLOUCESTER, ENGLAND
* CODE: 303
* DESC.: BIRMINGHAM, ENGLAND
* CODE: 304
* DESC.: WARRINGTON, ENGLAND
* CODE: 101
* DESC.: TORONTO
* CODE: 501
* DESC.: DUSSELDORF, GERMANY
* CODE: 1201
* DESC.: STOCKHOLM, SWEDEN
* CODE: STOP
* UPDATED.
```

Now I want to list the code-description table for *LOCATION*. I want to sort the table in ascending numeric sequence by code.

```
02:LC
* LIST CODES/DESCS.
* FOR WHICH FIELD: LOCATION
* SORT BY CODE OR DESC. (C/D): C
* ALIGN PAPER, TYPE SPACE, CR TO START.

* CURRENT NUMBER OF CODES IS 7.
* SUGGESTED MAXIMUM NUMBER OF CODES IS 100.

0101  TORONTO
0301  LONDON, ENGLAND
0302  GLOUCESTER, ENGLAND
0303  BIRMINGHAM, ENGLAND
0304  WARRINGTON, ENGLAND
0501  DUSSELDORF, GERMANY
1201  STOCKHOLM, SWEDEN

* FOR WHICH FIELD: STOP
```

Below, I list all the records in the system. I want to print all the fields, plus the descriptions of *LOCATION*. I want to sort the records in alphabetical order by *MAILBOX*. Note that no country exists for code 0101, and that no descriptions exist for *LOCATION* values 0201 and 1101. The blank descriptions for values 0201 and 1101 resulted because I was called out for a short beer and forgot to add them to the code-description table during the *AC* entry above.

```
02:LR
* LIST RECORDS
* CONSTRAINTS: ALL
   14 FOUND.
* PRINT FIELDS: ALL,(LOCATION)
* SORT FIELDS: MAILBOX
* ALIGN PAPER, TYPE SPACE, CR TO START.

MAILBOX LOCATION (LOCATION)

BJP          0303 BIRMINGHAM, ENGLAND
DCO          0303 BIRMINGHAM, ENGLAND
DINO         0302 GLOUCESTER, ENGLAND
DPM          0101 TORONTO
GHR          0501 DUSSELDORF, GERMANY
HCA          1201 STOCKHOLM, SWEDEN
HHY          0101 TORONTO
JMC          0501 DUSSELDORF, GERMANY
JMS          0101 TORONTO
KMC          0301 LONDON, ENGLAND
NICKT        0304 WARRINGTON, ENGLAND
RAJ          0301 LONDON, ENGLAND
RSG          0201
SJT          1101

* CONSTRAINTS: STOP
```

Now I want to make changes to *LOCATION*'s code-description table. I overwrite the description of code 0101 so that it says *TORONTO, CANADA*; I enter the descriptions for the blank values at 0201 and 1101 in "(*LOCATION*)", as displayed above during *LR*.

```
02:AC
* ADD CODES/DESCS.
* FOR WHICH FIELD: LOCATION
* CURRENT NUMBER OF CODES IS 7.
* MAXIMUM NUMBER OF CODES IS 100.
* CODE: 101
* DESC.: TORONTO, CANADA
* CODE: 201
* DESC.: ROCHESTER, U.S.A.
* CODE: 1101
* DESC.: COPENHAGEN, DENMARK
* CODE: STOP
* UPDATED.
```

Here, I want to list the code-description table again. Notice that I have nine code values now, instead of seven.

```
02:LC
* LIST CODES/DESCS.
* FOR WHICH FIELD: LOCATION
* SORT BY CODE OR DESC. (C/D): C
* ALIGN PAPER, TYPE SPACE, CR TO START.
* CURRENT NUMBER OF CODES IS 9.
* SUGGESTED MAXIMUM NUMBER OF CODES IS 100.

0101  TORONTO, CANADA
0201  ROCHESTER, U.S.A.
0301  LONDON, ENGLAND
0302  GLOUCESTER, ENGLAND
0303  BIRMINGHAM, ENGLAND
0304  WARRINGTON, ENGLAND
0501  DUSSELDORF, GERMANY
1101  COPENHAGEN, DENMARK
1201  STOCKHOLM, SWEDEN

* FOR WHICH FIELD: STOP
```

Now I want to list all the records again, this time with the *LOCATION* descriptions first. I want to sort in alphabetical order by *MAILBOX*.

```
02:LR
* LIST RECORDS
* CONSTRAINTS: ALL
  14 FOUND.
* PRINT FIELDS: (LOCATION),MAILBOX,LOCATION
* SORT FIELDS: MAILBOX
* ALIGN PAPER, TYPE SPACE, CR TO START.

(LOCATION)           MAILBOX LOCATION
BIRMINGHAM, ENGLAND  BJP         0303
BIRMINGHAM, ENGLAND  DCO         0303
GLOUCESTER, ENGLAND  DINO        0302
TORONTO, CANADA      DPM         0101
DUSSELDORF, GERMANY  GHR         0501
STOCKHOLM, SWEDEN    HCA         1201
TORONTO, CANADA      HHY         0101
DUSSELDORF, GERMANY  JMC         0501
TORONTO, CANADA      JMS         0101
LONDON, ENGLAND      KMC         0301
WARRINGTON, ENGLAND  NICKT       0304
LONDON, ENGLAND      RAJ         0301
ROCHESTER, U.S.A.    RSG         0201
COPENHAGEN, DENMARK  SJT         1101

* CONSTRAINTS: STOP
```

I'm finished with my work now, so I decide to exit MABRA.

```
02:STOP
* SYSTEM MODE
:STOP
```

4. How Multi-User MABRA Works

This section shows how MABRA works for multiple users. My user number is 2345678. I intend to set access so that user 3456789 can use one of my MABRA systems. First, I load the appropriate MABRA workspace and start the MABRA program. I want to enter MABRA system number 1, *ACME COMPUTER SERVICES.*

```
      )LOAD 86 MABRA
SAVED 17.53.43  06/07/79
      START
* MABRA. VERSION 2.
* SYSTEM MODE
:ENTER
* SYSTEM NUMBER: 1
* DATA MODE, SYSTEM IS 01: ACME COMPUTER SERVICES
```

Now I list the field definitions to see the structure of the fields.

```
01:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.

             01: ACME COMPUTER SERVICES     DATE: 23/10/79

NAME             TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

SURNAME          CHAR    10      Y       ACL              10A1
DOBIRTH          DATE     1      Y       ACL              1G⎕ZZ/ZZ/Z9⎕
SEX              CHAR     1      Y       ACL              1A1
DOJOIN           DATE     1              ACL              1G⎕ZZ/ZZ/Z9⎕
SALARY           REAL     1              ACL              CI7
LOCATION         CHAR     4      Y       ACL       25     4A1
```

I can find out how many records are in the system by doing the following:

```
01:LR
* LIST RECORDS
* CONSTRAINTS: ALL
   52 FOUND.
* PRINT FIELDS: STOP
* CONSTRAINTS: STOP
```

I'm ready to set access for user 3456789. I enter the command access, record access, and field access that I want to give. Regarding field access, I have to specify which fields user 3456789 can access when adding, changing, and listing records.

```
01:SA
* SET ACCESS
* USER(S): 3456789
* USERS ARE 3456789. (Y/N): Y
* COMMAND ACCESS
* COMMANDS: LR,LF,LC,AN,AR,CR,CM
* RECORD ACCESS
* CONSTRAINTS: LOC=01
   14 FOUND.
* OK? (Y/N): Y
* FIELD ACCESS
* AR FIELD ACCESS
* FIELDS: ALL,~SALARY
* MORE? (Y/N): N
* CR FIELD ACCESS
* FIELDS: LOCATION
* MORE? (Y/N): N
* LR FIELD ACCESS
* FIELDS: ALL,~SALARY
* MORE? (Y/N): N
* SET ACCESS? (Y/N): Y
* ACCESS SET.
* USER(S): STOP
```

After setting access, I want to check it by listing access. At the *USER(S)* prompt, I enter the number that I want to check. Then, MABRA displays the combinations of command, record, and field access that I gave user number 3456789.

```
01:LA
* LIST ACCESS
* USER(S): 3456789
* ALIGN PAPER, TYPE SPACE, CR TO START.

* USER IS 3456789.

  COMMANDS: LR,LF,CR,AR,AN,LC,CM

  RECORD CONSTRAINT: LOC=01

  FIELD               ADD CHA LIS
  SURNAME              X       X
  DOBIRTH              X       X
  SEX                  X       X
  DOJOIN               X       X
  LOCATION             X   X   X
--------------------------------
```

I'm finished setting access, so now I want to exit MABRA, sign off the APL system, and let user 3456789 sign on and use my terminal.

```
01:STOP
* SYSTEM MODE
:STOP
      )OFF HOLD
```

Below, user 3456789 signs on the APL system to use my MABRA system. I'll turn the rest of the demonstration over to him.

```
      )3456789:lock
   6461) 14.38.20  10/17/79 NESLEUTH

 SHARP APL SERVICE
      )LOAD 86 MABRA
 SAVED 17.53.43  06/07/79
```

Because I want to work with a MABRA system that I do not own (but have valid access to) I must assign the owner's user number (2345678) as the *CREATOR*. Then I can start the MABRA program.

```
      CREATOR←2345678
      START
* MABRA. VERSION 2.
* SYSTEM MODE
```

I want to list the systems to which I have access.

```
:LIST
* LIST SYSTEM(S)
* ALIGN PAPER, TYPE SPACE, CR TO START.

MABRA SYSTEMS ON USER: 2345678  DATE: 23/10/79

NO.  NAME

01  ACME COMPUTER SERVICES

* SYSTEM MODE
```

Now I enter the system.

```
:ENTER
* SYSTEM NUMBER: 1
* DATA MODE, SYSTEM IS 01: ACME COMPUTER SERVICES
```

By doing the following, I can see all the commands to which I have access.

```
01:HELP
ENTER ONE OF: LR;LF;CR;AR;AN;LC;CM;HELP;STOP
01:HELP
ENTER ONE OF:
        LR       - LIST RECORDS WITHIN CERTAIN CONSTRAINTS
        LF       - LIST FIELD NAMES AND DESCRIPTIONS
        CR       - CHANGE RECORD INFORMATION
        AR       - ADD NEW RECORDS
        AN       - FREQUENCY ANALYSIS (CROSSTABULATION)
        LC       - LIST CODE-DESCRIPTION TABLE
        CM       - CHANGE MULTIPLE RECORDS
        STOP     - RETURN TO 'SYSTEM MODE'
```

I want to list the field definitions to see the field structure. Notice that there is no *SALARY* field in my version and that the only field I can change is *LOCATION*.

```
01:LF
* LIST FIELD
* ALIGN PAPER, TYPE SPACE, CR TO START.

          01: ACME COMPUTER SERVICES     DATE: 23/10/79

NAME             TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

SURNAME          CHAR   10      Y      AL               10A1
DOBIRTH          DATE    1      Y      AL               1G⎕ZZ/ZZ/Z9⎕
SEX              CHAR    1      Y      AL               1A1
DOJOIN           DATE    1             AL               1G⎕ZZ/ZZ/Z9⎕
LOCATION         CHAR    4      Y      ACL       25     4A1
```

When I list all the records in my version, I can only access those in *LOCATION*=01.

```
01:LR
* LIST RECORDS
* CONSTRAINTS: ALL
   14 FOUND.
* PRINT FIELDS: ALL
* SORT FIELDS: SURNAME
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME      DOBIRTH SEX   DOJOIN LOCATION


ALLEN         8/05/28 F   15/12/42 0102
BROWN         1/03/20 F    6/06/35 0104
CHAUHAN      12/08/56 M    1/08/78 0110
COOPER       18/06/29 F   20/08/49 0101
FELL         16/08/23 M    1/08/37 0101
HUTCHINSON   15/05/51 M   13/09/69 0101
KILMISTER    11/01/22 F    7/01/39 0103
NORLEDGE     14/01/44 M   18/02/65 0101
PHILLIPS     30/05/35 M    1/02/69 0110
PRICE         6/10/25 M    7/11/53 0110
ROBERTSON     8/09/48 F   21/02/77 0101
SMITH         1/09/21 M   27/01/58 0102
TROOP        21/11/47 M    3/01/72 0110
WALKER       31/01/35 F   18/02/66 0102
```

If I try to find records with *SALARY*=10000, my version doesn't recognize *SALARY* as a valid field.

```
* CONSTRAINTS: SALARY=10000
? IDENTIFIER UNDEFINED
SALARY=10000
^
```

If I try to list the names, dates of birth, and salaries of all females in my version, I'm told I can't access *SALARY*. However, I can proceed with valid fields.

```
* CONSTRAINTS: SEX=F
   6 FOUND.
* PRINT FIELDS: SURNAME,DOBIRTH,SALARY
?'SALARY' IS AN INVALID FIELD NAME.
* PRINT FIELDS: SURNAME,αDOBIRTH
* SORT FIELDS: DOBIRTH
* ALIGN PAPER, TYPE SPACE, CR TO START.

SURNAME     αDOBIRTH

BROWN              59
KILMISTER          57
ALLEN              51
COOPER             50
WALKER             44
ROBERTSON          31

* CONSTRAINTS: STOP
```

I try to delete a field but am told that I don't have the access to do so.

```
01:DF
? INVALID OPTION
```

I have one record to add to the system, and can do this as demonstrated below.

```
01:AR
* ADD RECORD

SURNAME  :JONES
DOBIRTH  :031026
SEX      :F
DOJOIN   :240759
LOCATION :0103

SURNAME  :STOP
* UPDATE REQUEST NO. 3 FILED.
```

If I try to update my version with the record I just added, I find I can't do this because I don't have *UP* access.

```
01:UP
? INVALID OPTION
```

Now that I have added the record within my domain to the system, and have gotten an idea about the commands, records, and fields that I can access, I'm finished with the system. I exit user 2345678's MABRA system by entering *STOP* after the command prompt (`01:`) in Data Mode, and again, after the command prompt (`:`) in System Mode.

```
01:STOP
* SYSTEM MODE
:STOP
```

THE UTILITY WORKSPACE - 85 *MABRAUTIL*

This workspace provides programmers with a non-interactive method for reading and updating records in a MABRA system. As a programmer, you can use 85 *MABRAUTIL* to devise a system application for special prompts or special data processing. You can also use 85 *MABRAUTIL* to interact with another MABRA system.

85 *MABRAUTIL* contains the following functions:

1. *ENTER*	- enter a MABRA system
2. *SEARCHFOR*	- search records for a constraint
3. *READ*	- read records into the workspace
4. *APPEND*	- append (add) records to the system
5. *REPLACE*	- update records
6. *DELETE*	- delete records
7. *EXIT*	- exit a MABRA system
8. *GETDESC*	- retrieve descriptions from the code-description table
9. *GRADE*	- sort records in the workspace
10. *REORD*	- reorder records in the workspace
11. *DDMMYY*	- convert date fields from the YYMMDD format to DDMMYY

The following APL session demonstrates how to use workspace 85 *MABRAUTIL* to access a MABRA system and execute various commands. However, before getting into 85 *MABRAUTIL*, the APL session uses 86 *MABRA* to enter the particular MABRA system, 53: *DEMONSTRATION PERSONNEL*, and to display its field definitions and its records.

```
      )LOAD 86 MABRA
SAVED  16.01.21 10/09/79

      CREATOR←1234567
      START

* MABRA. VERSION 2.
* SYSTEM MODE
:ENTER◇53

* DATA MODE, SYSTEM IS 53: DEMONSTRATION PERSONNEL
53:LF◇ ◇
            53: DEMONSTRATION PERSONNEL     DATE:  9/10/79

NAME            TYPE WIDTH  INVERTED  ACCESS DESC-WID  ⎕FMT

SURNAME         CHAR   15             ACL              15A1
INITIALS        CHAR   3              ACL              3A1
SALARY          INT    1              ACL              CI6
SEX             CHAR   1              ACL              1A1
DOBIRTH         DATE   1              ACL              1G⎕ZZ/ZZ/Z9⎕
LOC             CHAR   2              ACL       20     2A1

53:LC◇LOC◇C◇ ◇ ◇STOP

* CURRENT NUMBER OF CODES: 4
* SUGGESTED MAXIMUM NUMBER OF CODES: 200

CV  VANCOUVER, CANADA
EC  COVENTRY, ENGLAND
FP  PARIS, FRANCE
UD  DALLAS, U.S.A.

53:LR◇ALL◇Δ,ALL,(LOC)◇ ◇ ◇ ◇STOP

Δ SURNAME         INITIALS SALARY SEX  DOBIRTH LOC (LOC)

1 SMITH           CC       15,000 F    1/04/52 CV  VANCOUVER, CANADA
2 MCDUFF          DR       18,000 F    3/12/49 EC  COVENTRY, ENGLAND
3 JONES           HR       14,296 M    7/10/55 UD  DALLAS, U.S.A.
4 SMITH           DN       16,000 M   19/11/41 FP  PARIS, FRANCE
5 CRAIG           JF       19,451 M   10/12/26 EC  COVENTRY, ENGLAND
6 BLACK           JB       16,721 M   27/06/51 UD  DALLAS, U.S.A.
7 CLARKE          JF       19,451 M   25/12/50 EC  COVENTRY, ENGLAND
8 MAY             BJ       24,000 F    6/03/54 CV  VANCOUVER, CANADA

53:QUIT
```

The functions from 85 *MABRAUTIL* are demonstrated below, still using the MABRA system, 53: *DEMONSTRATION PERSONNEL*.

1. Entering a MABRA System - *ENTER*

The first thing you must do in 85 *MABRAUTIL* is to call the function *ENTER* so you can reach the MABRA system you wish to work with. The *ENTER* function in 85 *MABRAUTIL* works in the same way as the *ENTER* command in 86 *MABRA*. When called, the function *ENTER* ties appropriate files and defines global variables required by the other functions. The syntax of *ENTER* is,

ENTER **system-number**

where **system-number** represents the system number of the system you wish to enter. As in 86 *MABRA*, a variable named *CREATOR* (the owner of the system) needs to be defined prior to calling *ENTER*. If *CREATOR* is given no value, then ⎕*AI*[1] is assumed.

The *ENTER* function ties the following files, described in the chapter, "File Documentation".

File Identification	**Description**
'creator *MAS***xx'**	master file
'creator *INV***xx'**	inverted fields file
'creator *DIR***xx'**	directory file
'user *MSK***xx'**	mask file (owned by user: 1↑⎕*AI*)

Referring to the file identifications above, **creator** represents the *CREATOR*'s user number; **xx** represents the system number (right justified and zero-filled); **user** represents a valid user's user number. *ENTER* also ties the MABRA function file, residing in library 86 and containing all the MABRA functions.

ENTER defines the following global variables, also described in the chapter, "File Documentation".

Variables	Contents
FLDNAMES	- field names
FLDFMTS	- field ⎕*FMT* arguments
FLDDESCS	- field descriptions
NUMRCDS	- number of records in system
MASFSZ	- number of components in master file
MASBLKSIZE	- number of records per master component
MAXNUMRCDS	- maximum number of records allowed in system
∆FLDNAMES	- names of variables in master file
MSKTY	- mask file's tie number
DIRTY	- directory file's tie number
INVTY	- inverted file's tie number
MASTY	- master file's tie number
FNTY	- tie number of MABRA function file
CURSYSNAME	- current system name
CCURSYSNUM	- current system number

The following event numbers allow you to trap error conditions:

Event		Indicates
507	-	no MABRA systems exist on user number defined by *CREATOR*
508	-	the MABRA system requested does not exist
509	-	problems exist with accession of records because the record access constraint is invalid

Example

```
      )LOAD 85 MABRAUTIL
SAVED  18.55.00 09/11/79

      CREATOR←1234567
      ENTER 53
      FLDNAMES,FLDFMTS,⍕FLDDESCS
SURNAME        15A1            0 15  0  0 15  0  0
INITIALS       3A1             0  3  0  0  3  0  0
SALARY         CI6             1  1  0  0  6  0  0
SEX            1A1             0  1  0  0  1  0  0
DOBIRTH        1G⎕ZZ/ZZ/Z9⎕    4  1  0  0  8  0  0
LOC            2A1             0  2  0  0  2 20 21
      ⎕NAMES
        86 MABRAV2
   2629674 MAS53
   2629674 INV53
   2629674 DIR53
   2629674 MSK53
      ⎕NUMS
1 2 3 4 5
      NUMRCDS
8
```

2. Searching Records - *SEARCHFOR*

SEARCHFOR identifies records within the current system in the same manner as the *CONSTRAINTS* prompt identifies records in 86 *MABRA*. In fact, 85 *MABRAUTIL* uses the same *SEARCHFOR* function as does 86 *MABRA*. The syntax of *SEARCHFOR* is,

R←SEARCHFOR **constraint**

where **constraint** represents a valid MABRA constraint (as described in the MABRA Constraints section of the chapter, "Data Mode"). The explicit result *R* indicates that the constraint is valid (0) or invalid (1), and **not** that the search identifies a number of records.

If **constraint** is valid (*R*=0), *SEARCHFOR* defines the following global variables:

FOUNDMASK - is a binary vector of length equal to *NUMRCDS*. *FOUNDMASK* indicates which records are identified by the constraint. A value of 1 in position **i** of *FOUNDMASK* indicates that record number **i** satisfies the constraint.

READPTR - is set to 0, and is used in retrieving records (see *READ*)

REPLACEPTR - is set to 0, and is used in updating records (see *REPLACE*)

Examples

```
      SEARCHFOR 'SURNAME=SMITH'
0
      +/FOUNDMASK
2                              (2 found)

      SEARCHFOR 'SALARY=ABC'
? VALUE TYPE INCOMPATIBLE WITH FIELD TYPE       (output generated by SEARCHFOR)
SALARY=ABC
       ^
1                              (invalid result of SEARCHFOR)
```

3. Retrieving Records - *READ*

READ retrieves records into the workspace. The syntax of *READ* is,

*R←***num-recs** *READ* **field-names**

where **num-recs** represents the number of records to be read, and **field-names** represents a list of field names, separated by commas, or represents *ALL*, or represents indices into *FLDNAMES*.

The records to be read are identified by the global variables *FOUNDMASK* and *READPTR*. As mentioned above, *FOUNDMASK* identifies the positions of the records, where *FOUNDMASK/⍳⍴FOUNDMASK* are the actual record numbers. *READPTR* is an index into *FOUNDMASK* indicating the position of the last record read in.

The left argument **num-recs** may be omitted. In this case a value for **num-recs** is calculated based on *⎕WA* (workspace available). You should choose the value of **num-recs** carefully, because reading in too few records at one time may result in inefficient processing; reading in too many records may cause a *WS FULL* error.

The explicit result *R* is the actual number of records retrieved. If *R* equals zero, there are no more records to be read, indicating the end of the file.

READ stores the retrieved records by field name, in global variables of rank 2 (that is, matrices). The rows of each of these variables correspond to one another, meaning the ith row of each variable represents data for the ith record. The naming convention used for these variables consists of prefixing the name of the field with a delta (∆).

The global variable, *READPTR*, is updated after every *READ*, indicating the position of the last record read in. By executing *READ* successively, you read records into the workspace until the end of the file is reached. *READPTR* should be reset to zero before newly identified records are to be read (this is done automatically by *SEARCHFOR*).

The following event numbers allow you to trap certain error conditions:

Event	**Indicates**
504	- invalid indices into *FLDNAMES*
505	- invalid field names
506	- ambiguous field names

Example 1

```
      SEARCHFOR 'SURNAME=SMITH'
0
      +/FOUNDMASK
2
      READ 'INITIALS,SALARY'
2
      ∆INITIALS
CC
DN
      ∆SALARY
15000
16000
      READ 'INITIALS,SALARY'          (later callings of READ recreate the ∆ variables)
0                                     (end of the file is reached)
```

Example 2

```
      FOUNDMASK←NUMRCDSρ0
      FOUNDMASK[1 3 6]←1              (identify 1st, 3rd, and 6th records)
      READPTR←0
      10 READ 1 3 4                   (read SURNAME, SEX, DOBIRTH)
3
      ∆SURNAME,∆SEX,⍕∆DOBIRTH
SMITH    F  520401
JONES    M  551007
BLACK    M  510627
      READPTR
4
```

4. Add New Records - *APPEND*

APPEND adds new records to the current system. The syntax of *APPEND* is,

R←APPEND **field-names**

where **field-names** represents a list of field names, *ALL*, or indices into *FLDNAMES*.

The field values used by *APPEND* must be supplied in global variables of rank 2. The naming convention for these variables is the same as for the result of *READ* (you prefix the field name with a Δ). Each variable **must** contain the correct data type (character, integer, etc.) and width, because *APPEND* does **not** check these. Each variable must also contain the same number of rows. Before *APPEND* adds records to the system, those fields not listed in **field-names** are given values of either blanks or zeros, depending on the data type.

The result *R* contains the number of records that *APPEND* adds to the current system.

The following event numbers allow you to trap certain error conditions:

Event	**Indicates**
501	- field values stored in Δ variables do not have the same number of rows
502	- maximum number of records allowed in system will be exceeded
504	- invalid indices into *FLDNAMES*
505	- invalid field names
506	- ambiguous field names

Example 1

```
      ΔSURNAME←1 15ρ15↑'MCDONALD'
      ΔINITIALS←1 3ρ'OLD'
      APPEND 'SURNAME,INITIALS'
1
```

Example 2

Assume the field values for the records being added are stored in the variable, *BLK* (ρ=4 32).

```
      ΔSURNAME←BLK[;ι15]
      ΔINITIALS←BLK[;15+ι3]
      ΔSEX←BLK[;,19]
      ΔDOBIRTH←((1↑ρBLK),1)ρ⎕FI,BLK[;19+ι6],' '
      ΔSALARY←((1↑ρBLK),1)ρ⎕FI,BLK[;25+ι5],' '
      ΔLOCATION←BLK[;31 32]
      APPEND 'ALL'
4
```

5. Updating Records - *REPLACE*

REPLACE updates the field values of records in the same way that the *CD* command does in 86 *MABRA*. The syntax of *REPLACE* is,

R←REPLACE **field-names**

where **field-names** represents a list of field names, *ALL*, or indices into *FLDNAMES*.

The records to be updated are identified by the global variables, *REPLACEPTR* and *FOUNDMASK*. As mentioned previously, *FOUNDMASK* identifies the position of the records being updated. *REPLACEPTR* is an index in *FOUNDMASK* indicating where the update starts. In most cases, *REPLACEPTR* should be set to zero initially (this is done automatically by *SEARCHFOR*).

The data used to update the records must be supplied in global variables of rank 2, having as names those of the field names prefixed with a Δ. These variables **must** have the same number of rows, and **must** have the correct data type and width. If the number of records identified in *FOUNDMASK* exceeds the number of rows (which we will call **i**) in the Δ variables, then only the first **i** records are updated. If **i** exceeds the number of records identified (which we will call **j**), then only the first **j** rows of the Δ variables are used.

The explicit result *R* contains the number of records updated. The rows of the Δ variables used for the update are dropped.

The global variable, *REPLACEPTR*, is updated after every *REPLACE* to indicate the position of the last record updated. Therefore, successive replaces update successive records until all records identified are updated.

The following event numbers allow you to trap certain error conditions:

Event	**Indicates**
504	- invalid indices into *FLDNAMES*
505	- invalid field names
506	- ambiguous field names

Example 1

```
      SEARCHFOR 'LOCATION=EC'
0
      +/FOUNDMASK
3
      READ 'SALARY'
3
      ΔSALARY←⌊.5+1.1×ΔSALARY
      REPLACE 'SALARY'
3
```

Example 2

The following function is appropriate for a system that has more records than *READ* and *REPLACE* can process in one execution. For example, if you read a large number of records into the workspace, and the size of each block of records is 20 kilobytes, you may get a *WS FULL* error. However, by not supplying a left argument to *READ*, you allow MABRA to choose an appropriate number of records to fit in the workspace.

```
    ∇ TENPERC;T
[1]   T←SEARCHFOR 'LOCATION=EC'
[2]   (+/FOUNDMASK);' FOUND'
[3]   LOOP:→(0=READ 'SALARY')/END
[4]   ∆SALARY←⌊.5+1.1×∆SALARY
[5]   T←REPLACE 'SALARY'
[6]   →LOOP
[7]   END: 'UPDATE DONE'
    ∇
```

Example 3

This example demonstrates a type of *SALARY* update whereby the updates are stored in the variable *BLK*, such that the first four columns of *BLK* represent the absolute record number of the record being changed and the remaining 6 columns represent the new salary.

```
        ⍴BLK
15 10
        RCDNUMS←⎕FI,' ',BLK[;⍳4]
        ∆SALARY←⎕FI,' ',0 4↓BLK
        IND←⍋RCDNUMS
        ∆SALARY←((⍴∆SALARY),1)⍴∆SALARY[IND]
        FOUNDMASK←NUMRCDS⍴0
        FOUNDMASK [RCDNUMS]←1
        REPLACEPTR←0
        REPLACE 'SALARY'
15
```

6. Deleting Records - *DELETE*

DELETE removes specified records from the current system. Such records are not **physically** removed from the system but are flagged (re-defined) as deleted. To permanently remove "deleted" records from a system, you must use the *RB* (reblock) command in 86 *MABRA*. *DELETE* has the following syntax,

```
        R←DELETE
```

where the explicit result *R* contains the number of records that *DELETE* flags as unattainable. Values of 1 in the global variable *FOUNDMASK* identify records for deletion. As mentioned earlier, *FOUNDMASK* is a binary vector of length equal to *NUMRCDS*.

Example

```
        SEARCHFOR 'LOCATION=EC'
0
        +/FOUNDMASK
3
        DELETE
3
        SEARCHFOR 'LOCATION=EC'
0
        +/FOUNDMASK
0
```

7. Exiting the Current System - *EXIT*

EXIT enables you to leave the current system. You must exit one MABRA system before you can enter another. Exiting a system involves untying the files associated with the system and resetting the global variables defined by *ENTER*. The syntax of *EXIT* is:

```
EXIT
```

8. Getting Descriptions From Code-Description Tables - *GETDESC*

When given a list of codes, *GETDESC* retrieves a coded field's descriptions. The syntax of *GETDESC* is,

R←**field-name** *GETDESC* **codes**

where **field-name** represents a field name, or is an index into *FLDNAMES*, and **codes** represents a matrix of codes for which descriptions are desired.

The explicit result *R* contains descriptions which correspond to **codes**. If a code is not found in the code-description table, a blank "description" results.

Recall the code-description table for the *LOCATION* field displayed in the APL session, at the beginning of this chapter:

Code **Description**

```
 CV   - VANCOUVER, CANADA
 EC   - COVENTRY, ENGLAND
 FP   - PARIS, FRANCE
 UD   - DALLAS, U.S.A.
```

Example 1

```
      'LOCATION' GETDESC 3 2ρ'ECCTEC'
COVENTRY, ENGLAND
TORONTO, CANADA
COVENTRY, ENGLAND
```

Example 2

```
      SEARCHFOR 'SURNAME=SMITH'
      READ 'SURNAME,INITIALS,LOCATION'
2
      '15A1,X1,3A1,X1,20A1' ⎕FMT (∆SURNAME;∆INITIALS;'LOCATION' GETDESC ∆LOCATION)
SMITH          CC   VANCOUVER, CANADA
SMITH          DN   PARIS, FRANCE
```

9. Sorting Records in the Workspace - *GRADE*

In a MABRA system, records are stored in the order in which they were entered. Once records are retrieved into the workspace (using *READ*), they may require sorting. The function *GRADE* returns indices by which records are selected for sorting. The syntax of *GRADE* is,

R←GRADE **field-names**

where **field-names** represents a list of sort fields (as in the *LR* command in `86` *MABRA*), or **field-names** represents indices into *FLDNAMES*, where a positive index indicates ascending order and a negative index indicates descending order. If more than one field is specified in **field-names**, the leftmost field is the major sort field.

The global variables used by *GRADE* are the field values stored in the ∆ variables. The ∆ variables must have the same number of rows.

The explicit result *R* contains an integer vector of indices into the records in the workspace. These records, when indexed by *R*, are reordered into the sorted order that you requested with *GRADE*.

The following event numbers allow you to trap certain error conditions:

Event	**Indicates**
504	- invalid indices into *FLDNAMES*
505	- invalid field names
506	- ambiguous field names

Example

```
      ∆SURNAME,∆INITIALS,⍕∆SALARY
SMITH       CC    15000
JONES       HR    14296
BLACK       JB    16721
      GRADE 'SURNAME'
3 2 1
      GRADE '-SALARY'
3 1 2
      GRADE 2 1
1 2 3
```

10. Reordering Records in the Workspace - *REORD*

REORD is generally used with the function *GRADE*. Using indices that *GRADE* produces, *REORD* sorts the records accordingly in the workspace. The syntax of *REORD* is,

field-names *REORD* **indices**

where **field-names** represents a list of field names (or indices into *FLDNAMES*), and **indices** represents the indices into the records in the workspace.

The records are represented in the workspace by the ∆ global variables.

The following event numbers are used to allow you to trap certain error conditions:

Number	Indicates
504	- invalid indices into *FLDNAMES*
505	- invalid field names
506	- ambiguous field names
512	- a Δ variable is not a matrix
513	- the reordering index is out of range
514	- a Δ variable does not exist

Example

```
      ΔSURNAME,ΔINITIALS,⍕ΔSALARY
SMITH       CC   15000
JONES       HR   14296
BLACK       JB   16721

      'SURNAME,INITIALS,SALARY' REORD 3 1 2
      ΔSURNAME,ΔINITIALS,⍕ΔSALARY
BLACK       JB   16721
SMITH       CC   15000
JONES       HR   14296
```

11. Changing Dates into the DDMMYY Format - *DDMMYY*

MABRA stores dates in a year/month/day (YYMMDD) format. For example, August 12, 1959 is stored as `590812` (integer). Most applications require dates to print in day/month/year (DDMMYY) format. The *DDMMYY* function converts dates in YYMMDD format to DDMMYY. The syntax is,

R←DDMMYY **dates**

where **dates** represents numeric dates in the YYMMDD format, and *R* represents the converted result in DDMMYY format. The explicit result *R* has the same shape as **dates**.

Example

```
      ΔDOBIRTH
790101
550710
630721

      DDMMYY ΔDOBIRTH
010179
100755
210763
```

How to Change a Field's Data Type

Using the *CF* (change field) command, you can change all field definitions (name, width, inversion status, coded status, output format) **except** the **type** of data in a field. You, yourself, must change a field's type because such a change may require differences in representation which MABRA cannot predict. For example, when changing an **integer** type to a **real** type, you may decide to round the field values, truncate them, multiply them by some factor, etc.

To change a field's data type, you must work between a **source** field (the field whose type you wish to change) and a **target** field (the field containing the new type). A sample APL session, further below, shows how you go about this process, described immediately below:

1. `)LOAD 86 MABRA` and enter the *AF* command to add a new field to the system. This field is the **target** field for the new data type. When you define the target field during *AF*: enter a unique **temporary** field name; enter the **new** data type; enter the **new width**, based on the new type; opt for the new, default **output format** (or enter a new, alternative output format).

2. `)LOAD 85 MABRAUTIL`. Read the source field into the workspace. Convert the source field's data type to the appropriate type for the target field. Replace the source field with the target field.

3. `)LOAD 86 MABRA` and enter the *DF* command to delete the source field. Then, enter the *CF* command and change the **name** of the target field to the name of the deleted source field.

The following APL session shows how you can change a field's integer type to a character type having a width of four elements:

```
      )LOAD 86 MABRA
SAVED  16.01.21 10/09/79
      START
* MABRA  VERSION 2.
* SYSTEM MODE
:ENTER◇1
* DATA MODE, SYSTEM IS 01: EXAMPLE OF CHANGING FIELD'S TYPE
01:AF
* ADD FIELD
* ENTER FIELD NAME: NEW
* TYPE? (CHAR/INTE/REAL/BINA/DATE): CHAR
* NUMBER OF ELEMENTS IN FIELD: 4
* USE DEFAULT OUTPUT FORMAT? (Y/N): Y
* OK TO ADD? (Y/N): Y
* FIELD ADDED.
* ENTER FIELD NAME: QUIT
```

```
      )LOAD 85 MABRAUTIL
SAVED 18.02.05 11/16/79
      ∇ CHANGETYPE;T                    (here, you would define CHANGETYPE)
[1]     ENTER 1
[2]     T←SEARCHFOR 'ALL'
[3]     L1:→(0=READ 'LOCATION,NEW')ρL2
[4]     ΔNEW←'ZI4' ⎕FMT ΔLOCATION
[5]     T←REPLACE 'NEW'
[6]     →L1
[7]     L2:EXIT
[8] ∇
      CHANGETYPE                        (now, you would run CHANGETYPE)

      )LOAD 86 MABRA
SAVED 23.49.46 11/13/79
      START
* MABRA. VERSION 2.
* SYSTEM MODE
:ENTER◇1
* DATA MODE, SYSTEM IS 01: EXAMPLE OF CHANGING FIELD'S TYPE
01:DF
* DELETE FIELD(S)
* FIELDS TO BE DELETED: LOCATION
* DELETE LOCATION? (Y/N): Y
* DELETED.
* FIELDS TO BE DELETED: STOP
01:CF
* CHANGE FIELD
* FIELD NAME: NEW
* CHANGE FIELD=NEW
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT? NAME
* ENTER NEW FIELD NAME: LOCATION
* OK TO CHANGE NAME FROM 'NEW' TO 'LOCATION'? (Y/N): Y
* FIELD NAME CHANGED FROM 'NEW' TO 'LOCATION'.
* CHANGE FIELD=LOCATION
* CHANGE NAME, WIDTH, INVERT, CODE, ⎕FMT: QUIT
```

MABRA FILE DOCUMENTATION

Every MABRA system has three different levels of files associated with it:

- the first level consists of the *SYS* file—a directory of all MABRA systems that are owned by the *CREATOR*.
- the second level consists of the *DIR*, *MAS*, *INV*, *ACC*, and *UPD* files. The *CREATOR* owns these files. They contain information about a particular MABRA system.
- the third level consists of the *MSK* file. An individual user owns this file. This individual has valid access to one of the *CREATOR*'s MABRA systems.

First Level: the System Directory File - *SYS*

Every *CREATOR* of a MABRA system has a *SYS* file. This file contains information about all the systems that the *CREATOR* owns, and who can access these systems.

Component 1 contains a package of the following variables:

SYSNUMS - is an integer vector of system numbers that identify the systems that the *CREATOR* owns.

SYSNAMES - is a character matrix, ⍴*SYSNUMS* by 40, of system names that correspond to *SYSNUMS*.

SYSUSERS - is an integer vector of user numbers which have access to any of the MABRA systems owned by the *CREATOR*. *SYSUSERS*[1] represents the *CREATOR*.

SYSACC - is a binary matrix, ⍴*SYSNUMS* by ⍴*SYSUSERS*, indicating to which system each user has access. The expression:

`SYSACC[;SYSUSERS⍳x]/SYSNUMS`

indicates to which systems a user has access, where **x** represents a user number. Column 1 contains the *CREATOR*'s access, all elements consisting of the value 1.

Component 2 contains a package with the function *SUBMITREQUEST*.

SUBMITREQUEST - is a **locked** and **secure** function that submits N-task and B-task updates. *SUBMITREQUEST* is updated every time the *NEWPASS* command is used.

Second Level: the *DIR*, *MAS*, *INV*, *ACC*, and *UPD* Files

In practice, these files have a suffix on their file names: the **system number** of the particular MABRA system to which the file belongs. For example, the *DIR* file is really represented as *DIR***xx**, where **xx** is the MABRA system number. The system number is always right-justified and zero-filled, its limits being 01≤**xx**≤99.

1. The Miscellaneous Information File - *DIR*

The *DIR* file contains miscellaneous information about a particular MABRA system.

Component 1 contains a package of the variables described below.

FLDNAMES - is a character matrix of field names, **n** by 15, where **n** represents the number of fields, one per row. Column 15 is always blank.

FLDFMTS - is a character matrix, **n** by **m**, of a *⎕FMT* command string that formats a field. **n** represents the number of rows; **m**, the number of columns. The rows of *FLDFMTS* and *FLDNAMES* correspond.

FLDDESCS - is an integer matrix, **n** by 7, of numeric information for each field. Due to future enhancements, the number of columns may increase beyond 7.

[;1] references the field type as:
0 - character
1 - integer
2 - real
3 - binary
4 - date

[;2] references the field width (ρ).

[;3] references the file component number that starts the list of inverted fields in the *INV* file. If a field is not inverted, this number is 0.

[;4] references the number of rows in each block of an inverted field. The size of each block is about 20 kilobytes. This value is 0 if the field is not inverted.

[;5] references the number of output columns taken by the field after its output format (*FLDFMTS*) is calculated.

[;6] references the field's description width, if the field is defined as coded. The description width is 0 if the field is not coded.

[;7] references the file component number of *DIR*'s code-description table if the field is coded. The component number is ≥20 if the field is coded, 0 if not coded.

Component 2 contains an integer scalar of the number of records (*NUMRCDS*) currently in the system. The number of records also includes "deleted" records. The scalar is initially 0.

Component 3 contains a package containing the following variables:

MASBLKSIZE - is an integer scalar representing the blocking factor—the number of records held in each component—of *MAS*. The blocking factor is initially 200.

MAXNUMRCDS - is an integer scalar representing the maximum number of records allowed in the system. The limits are 1≤*MAXNUMRCDS*≤150000. You can change *MAXNUMRCDS* with the *RS* (resize) command.

Component 4 contains an integer vector, of length 4, containing update task parameters.

[1] represents 0 if the update task is free to run
1 if an N-task is pending
2 if a B-task is pending
3 if an N-task is active
4 if a B-task is active
5 if a file reservation error occurred
6 if the maximum number of records will be exceeded

[2] represents the file component number of the first update request pending in *UPD*.

[3] represents, if an update task is active, either the task-ID of the N-task, or the request number of the B-task. If an update task is not pending, this value is 0.

[4] indicates to the *CREATOR* that the file of completed update tasks (*UPD*) has reached its upper limit. At this point, the *CREATOR* can drop the lapsed (completed) components, or can increase the file's upper limit. This value is set initially to 50.

Component 5 contains a binary vector of a length equal to the number of records in the system. The number of elements equals the value of *DIR*'s component 2. The value 1 in any position of the vector indicates that the corresponding record in *MAS* is deleted. Such a deleted record remains in *MAS* until you purge it with the *RB* (reblock) command.

Component 6 contains a package of functions and variables that are used for field input validation. For more information, see the chapter, " Field Input Validation".

Component 7 contains the MABRA system's version number. This value is currently 2. Programmers should not change this value. If this value differs from the current version number, MABRA performs automatic file conversions to update your version.

Component 8 contains an integer vector of users (not including the *CREATOR*) currently using the system. This vector is set initially to ι0 and must always have a value of ι0 if critical commands such as *AR*, *CF*, *DF*, *RB*, *RS*, and *CS* are to be run. Note that if a user exits the system abnormally (through a break, line drop, or crash), his or her user number remains here until the user restarts the system.

Component 9 contains an integer scalar that indicates whether or not critical commands (see just above) can be run. If this value is zero, valid users can access the system. If this value is non-zero, this represents the *CREATOR*'s user number and indicates that no user can access the system until the completion of critical commands.

Component 10 is reserved for future enhancements.

Component 11	contains an integer matrix of indices into *FLDNAMES*. This matrix contains the various field orders for the *AR* command. Row 1 is always *ι1↑ρFLDNAMES*. A row containing *¯1* indicates that the particular field order is deleted.
Component 12 through Component 16	are reserved for future enhancements.
Component 17	contains a package of functions and variables that define related fields. For more information, see the chapter, " Related Fields".
Component 18	is reserved for future enhancements.
Component 19	is an integer vector of length 2 that contains the number of the update request in progress, and the number of records (*NUMRCDS*) in the system as the update task progresses. The update task reads this component if the task was suspended and needs to be restarted.
Component 20 and onward	are packages of variables that contain codes and descriptions of coded fields. Components are referenced by column 7 of *FLDDESCS*.

CODE -	contains a matrix of codes, **n** by *FLDDESCS*[**x**;2], with type *FLDDESCS*[**x**;1]. **x** represents the row number of *FLDDESCS* that contains the coded field.
DESC -	contains a character matrix, **n** by *FLDDESCS*[**x**;6], of code descriptions. **n** represents the number of rows. **x** represents the row number in *FLDDESCS* that contains the coded field. The rows of *DESC* and *CODE* correspond. Descriptions for a coded field are evoked during *LR*, *CR*, *DR*, and *BP* by enclosing a coded field in parentheses.
MAX -	contains an integer scalar representing the maximum number of codes allowable. MABRA sets this value during the *CF* (change field) command.

2. The Master File - *MAS*

This file contains all the records in a particular MABRA system. Each component of this file contains a package of variables which represent every field in the system. Each field is stored in a variable with shape *MASBLKSIZE* by *FLDDESCS*[**x**;2]. Each row holds a different record. **x** represents a field name that names the variable according to the following algorithm:

```
'⍙',FLDNAMES[x;]
```

The delta-underscore (⍙) prefixes the variable names in an attempt to eliminate naming conflicts within MABRA's software and the user's software.

Consider a system with two fields, *NAME* and *SEX*, and *MASBLKSIZE* equal to 50. To retrieve record number **n**, the following algorithm is used:

```
CN←⌈n÷MASBLKSIZE
ROW←1+MASBLKSIZE|n-1
⎕PDEF ⎕READ component CN of MAS
ΔNAME[ROW;];ΔSEX[ROW;]
```

Initially *MAS* is empty, but as soon as a field is added, a full component is appended, even if it contains no records. This is a means of avoiding problems with growing *⎕REPLACE*s.

3. The Inverted Field File - *INV*

This file represents a method for speeding up searches on commonly used fields. Rather than doing a slow sequential search through the *MAS* file (which is blocked at a low rate because other fields exist in the component), a very fast search can be applied to an inverted list (of rank 2) that is blocked at a high rate. Consider a sample system:

- number of records is 16,000
- *MASBLKSIZE* is 100
- our search field is, let's say, *SEX* which contains a one-character field

Case 1 - *SEX* field is not inverted
to find all males (*M*), 160 file reads are required

Case 2 - *SEX* field is inverted
block size of the inverted field is 16K
- to find all males (*M*), 1 file read is required

Inverting a field causes the field's data to be recorded a second time. You should evaluate this duplication of data when considering maintenance and storage expenses.

4. The Access File - *ACC*

This file contains information about users' access. A user's access comprises record, command, and field access. Component 1 contains an integer vector of user numbers, used as an index into the remaining components of the file. Each of these components contains a package of variables that define a single user's access:

CMDACC - is a binary vector that indicates (masks) all commands to which the user has access. A list of all the commands in MABRA is obtainable by using the algorithm,

```
'OPS' ⎕PVAL ⎕READ n,3
```

where **n** represents the file tie number of the MABRA function file in library 86.

RCDACC - is a MABRA constraint that identifies all records to which the user has access.

FLDACC - is a binary matrix, 3 by 1↑⍴*FLDNAMES*, of the user's field access. The rows represent **add**, **change**, and **list** access, respectively.

5. The Update Request File - *UPD*

This file contains all the update requests for a particular MABRA system. Each component represents a different update request. The component number is the same as the update request number.

UPD contains two classes of update requests: those which are pending release and those which have completed processing. Component 4 of the *DIR* file, discussed earlier in this chapter, contains the means for you to distinguish between processed and pending updates.

Each component also contains a package of variables that define the request. Each type of request (*AR*, *CR*, *CD*, *DR*, and *WU*) is represented as a different package, described below.

AR request package:
- *TYPE*←1
- *IND* is an integer vector of indices into *FLDNAMES* that specifies what field names were input.
- '*T⍙*',*FLDNAMES*[*IND*;] are the variables (rank 2) containing the input for each field and record.

CR request package:
- *TYPE*←2
- *CONS* is a character vector of the constraint that identifies the records to be changed.
- *FOUNDMASK* is a binary vector (ρ=*NUMRCDS*) indicating which records meet the *CONS* constraint.
- *FC* is an integer vector of indices into *FLDNAMES* indicating which fields are to be updated.
- '*T⍙*', *FLDNAMES*[*FC*;] are the variables (rank 1) containing the changes to field values.

DR request package:
- *TYPE*←3
- *CONS* is a character vector of the constraint that identifies records for deletion.
- *FOUNDMASK* is a binary vector (ρ=*NUMRCDS*) indicating which records meet the *CONS* constraint.

WU request package:
- *TYPE*←5
- *WU* changes a particular request's *TYPE* to 5.

CD request package:
- *TYPE*←6
- *IND* is an integer vector of indices into *FLDNAMES* that specifies what field names were entered.
- *CONS* is the same as in *CR*.
- *FOUNDMASK* is the same as in *CR*.
- (((ρ*IND*),2)ρ'*T⍙*'),*FLDNAMES* [*IND*;] are variables (rank 2) containing the changes made to field values.

Third Level: the Mask File - *MSK*

Every user who has valid access to a MABRA system owns a *MSK* file. *MSK* is a storage file for messages and temporary results from a user's MABRA session. *MSK* is usually of little importance after the MABRA session completes.

Component 1	contains a character vector of information about the status of batch prints (*BP*) and update tasks (*UP*). Whenever the *CREATOR* uses *ENTER* or *NS* (new snapshot), to enter Data Mode in 86 *MABRA*, component 1 of *MSK* prints any pending messages. Component 1 is then reset to ⍳0.
Component 2	contains a binary vector of length *NUMRCDS* that indicates which records the user can view, due to record deletions or access control.
Component 3	contains a binary vector of length *NUMRCDS* that indicates the records identified in the previous constraint. For more information, see *PREVIOUS* in the MABRA Constraints section of the chapter, "Data Mode".
Component 4	contains the results of the previous crosstabulation analysis (*AN*) because the analysis is run as an N-task, and the T-task needs the results for output.
Component 5 through Component 10	are reserved for future enhancements to MABRA.

RELATED FIELDS

In some MABRA systems certain fields relate to other fields. For example, a field, *PROFIT*, may relate to the fields, *SALES* and *EXPENSES*, where:

PROFIT←SALES-EXPENSES

Rather than entering a value for *PROFIT* every time *SALES* or *EXPENSES* is modified (through *AR*, *CR*, or *CD*), you can program MABRA to enter a value automatically, by giving MABRA the necessary information about field relations. The actual field relations are evaluated during the update (*UP*) N-task or B-task.

An Outline of the Update Task

Assume that an *AR*, *CR*, or *CD* update request updates more than one component of the *MAS* file.

LOOP: - loop for every *MAS* component to be updated

⎕READ - read, from the *MAS* file, appropriate fields to be updated

PREUPD - run this user-defined function to modify field data before making *AR*, *CR*, or *CD* changes

AR, *CR*, or *CD* - make field changes at this point

POSTUPD - run this user-defined function that modifies field data after the *AR*, *CR*, or *CD* changes have been made

⎕REPLACE - replace appropriate fields into the *MAS* component

→LOOP - loop to read in the next *MAS* component

Component 17 of the *DIR* File

Component 17 of the *DIR* file must contain a package of: the functions *PREUPD* and *POSTUPD*; the variable *BOOLREL*; other functions for performing the relation.

BOOLREL - is a binary array with ρ=3,2ρ1↑ρ*FLDNAMES* showing the related fields

[1;;] - is a plane that tells the update task what other fields need to be **read** in order to perform the relation. If fields **i** are to be updated, where **i** represent indices into *FLDNAMES*, then fields (∨/*BOOLREL*[1;;**i**])/ι1↑ρ*FLDNAMES* are to be read as well as fields **i**.

[2;;] - is a plane that tells the update task what other fields must be **written** into the *MAS* file. If fields **i** are to be updated, where **i** represent indices into *FLDNAMES*, then fields (∨/*BOOLREL*[2;;**i**])/ι1↑ρ*FLDNAMES* are to be written as well as fields **i**.

`[3;;]` - is a plane that calculates the right argument of the functions *PREUPD* and *POSTUPD*, indicating what relations to perform. If fields **i** are to be updated, then (∨/*BOOLREL*[3;;**i**])/ι1↑ρ*FLDNAMES* is passed as a right argument to *PREUPD* and *POSTUPD*.

The following example shows how *PREUPD*, *POSTUPD*, and *BOOLREL* are used in a system with the following fields:

Field Name	**Representation**
S	- sales (integer notation)
E	- expenses (integer notation)
P	- profit (integer notation)

The following array shows how to implement the relation, *P←S-E*.

```
           SEP
BOOLREL←   010   S
           100   E
           110   P

           000   S
           000   E
           110   P

           000   S
           000   E
           110   P
```

If *S* is updated through *AR*, *CR*, or *CD*, then:

- *E* and *P* (as well as *S*) are read because the value of *E* and the resultant field *P* are needed for *P←S-E*.
- *P* (as well as *S*) is written to file. There is no need to write *E* back to file because only its value is required, and its value has not changed.
- The right argument to *PREUPD* and *POSTUPD* is `,3` (meaning *P*).

Here is a definition of *POSTUPD*, based on the above application:

```
    ∇ POSTUPD A
[1]   →(A≠3)/0      ⍝ if A is 3 (P) then continue
[2]   ⍙P[TIND;]←⍙S[TIND;]-⍙E[TIND;]
    ∇
```

Remember that the field *P* is stored in each *MAS* file component as the variable ⍙*P*. *TIND* should **always** be used as the index to this field because *TIND* indicates which records are to be changed.

```
     ∇ PREUPD A
[1]    ⍝ this function is not needed for this example
     ∇
```

A Sample Session

For systems that require more than 1 relation, you can generalize *PREUPD* and *POSTUPD* in the following manner,

```
     ∇ POSTUPD A               (or PREUPD A)
[1]    A←(1=⎕NC 'L̲',FLDNAMES[A;])/A
[2]    LOOP: →(0∊⍴A)/0
[3]    ⍎'→L̲',FLDNAMES[A[1];]
[4]     L̲fieldnames: relation expressed in APL
[5]    →BLOOP
[6]    BLOOP:   A←1↓A   ◇   →LOOP
     ∇
```

so that in the relation, *P←S-E*, *PREUPD* can have the following lines inserted:

```
L̲P:    ⍙P[TIND;]←⍙S[TIND;] - ⍙E[TIND;]
       →BLOOP
```

The following example shows how to implement more than one relation in a larger system:

Field Name	Data Type
GROSS	integer
TAXRATE	real
NET	integer
TAX	integer
OLDNET	integer
DATE	date
SHIFT	binary

These are the relations that you might want to set up:

- *TAX←⌊.5+TAXRATE×GROSS*
  NET←GROSS-TAX

- *DATE←* date of last update to record

- if *SHIFT* is updated:
  OLDNET←NET
  NET←0

```
                    1234567

      BOOLREL←      0100000     1
                    1000000     2
                    1100001     3
                    1100000     4
                    0000001     5
                    1111111     6
                    0000000     7

                    0000000     1
                    0000000     2
                    1100001     3
                    1100000     4
                    0000001     5
                    1111111     6
                    0000000     7

                    1234567

                    0000000     1
                    0000000     2
                    1100000     3
                    0000000     4
                    0000000     5
                    1111111     6
                    0000001     7
```

```
      ∇ PREUPD A
[1]     A←(1=⎕NC 'L̲', FLDNAMES[A;])/A
[2]     LOOP: →(0=⍴A)/0
[3]     ⍎'→L̲', FLDNAMES [A[1];]
[4]     L̲DATE: ⍙DATE[TIND;1]←100⊥100|3↑⎕TS ⍝note that date is in YYMMDD format
[5]     →BLOOP
[6]     L̲SHIFT: ⍙OLDNET[TIND;]←⍙NET[TIND;]
[7]     ⍙NET[TIND;]←0
[8]     →BLOOP
[9]     BLOOP: A←1↓A ◇ →LOOP
      ∇

      ∇ POSTUPD A
[1]     A←(1=⎕NC 'L̲',FLDNAMES[A;])/A
[2]     LOOP: →(0=⍴A)/0
[3]     ⍎ '→L̲',FLDNAMES[A[1];]
[4]     L̲NET: ⍙TAX[TIND;]←⌊.5+⍙TAXRATE[TIND;]× ⍙GROSS[TIND;]
[5]     ⍙NET[TIND;]←⍙GROSS[TIND;]-⍙TAX[TIND;]
[6]     →BLOOP
[7]     BLOOP: A←1↓A ◇ →LOOP
      ∇
```

Notice that the shifting of *NET* to *OLDNET* must be done before the actual change of *NET* (see example). The setting of *DATE* can be done either in *PREUPD* or *POSTUPD*.

The following two utility functions are available in the update workspace and can be used by *PREUPD* and *POSTUPD*.

R←WHOUPD - returns the user number of the user who submitted the update request that is being processed currently

R←WHENUPD - returns a *⎕TS* that indicates the time of submission for the current update request, as opposed to the time of completion

Debugging Hints:

1. You should localize all variables that are used in *PREUPD* or *POSTUPD*; however, you should not localize *TIND*.

2. You can insert other functions or variables into component 17 for use with *PREUPD* and *POSTUPD*. The functions or variables, however, should not conflict with functions or variables already residing in the workspace.

3. When testing *PREUPD* and *POSTUPD*, you should monitor the N-task carefully by inquiring about *⎕RUNS*—to ensure that an infinite loop does not occur. If you think the N-task might be in an infinite loop, you merely need to *⎕BOUNCE* the N-task and do the following to restart the update as a T-task:

```
⎕SP←'DEBUG'
)LOAD  CUPDxx          (xx represents the system number, right-justified and zero-filled)
→⎕LC
```

FIELD INPUT VALIDATION

Validation of Data by Type

When entering field values with *AR*, *CR*, or *CD*, MABRA validates such values according to their types:

Type	Valid Input
character	any input is accepted
integer	input must be an integer from `-2*31` through `¯1+2*31`
real	any decimal number is accepted
binary	input values consisting of 1 or 0 are accepted
date	any date is accepted in the form DDMMYY, MMYY, YY, or 0 where DD≤31, MM≤12, YY≤99, and 0 usually indicates an unknown date

Often such basic validations are not sufficient for certain applications. For example, an integer field, *SALARY*, might require field values greater than 10000; a character field, *SURNAME*, might only allow alphabetic values.

MABRA lets you supply your own validation functions for values entered with *AR*, *CR*, and *CD*.

User-Definable Functions

You can define the following functions according to their accompanying guidelines. These functions must be ⎕*PACK*ed into component 6 of the *DIR* file.

1. *PREVAL*
   This function can initialize variables that are used in the actual validation functions, *VALIDATE* and *TINKER*. *PREVAL* is called at the beginning of the *AR*, *CR*, and *CD* commands.

2. *POSTVAL*
   This function can clean up the environment at the end of the *AR*, *CR*, or *CD* commands, before MABRA returns to the command prompt in Data Mode. *POSTVAL* is called at the end of the *AR*, *CR*, and *CD* commands.

3. *R←VALIDATE A*
   This function validates field input after it is converted to the proper type and width. The input is stored in the variable *INP*.

   The right argument *A* is an index into *FLDNAMES* that identifies *INP*'s field name. The result *R* must be one of the following:

Validation Indicator	Indication
0	an error was found during input
1	input is valid

4. *R←TINKER A*
   This function can modify and validate field values after ⍞ input, but before type conversion or simple MABRA validation occur. Field input is stored in the character vector *INP*.

   The right argument *A* is an index into *FLDNAMES*, specifying *INP*'s field name. The result *R* must be one of the following:

Validation Indicator	Indication		MABRA's Action
1	*INP* is valid so far	-	continue with MABRA type and width conversion; then execute the user-defined function *VALIDATE*
2	an error exists in *INP*	-	an error message prints, prompting for the field value represented by *INP*
3	*INP* is valid so far	-	MABRA does no further conversion or validation

Input Validation in a Sample System

Consider a system that contains information about airline pilots. The system contains the following fields:

Field Name		Content
SURNAME	-	pilot's last name
DOBIRTH	-	date of birth
AIRCRAFT	-	name of aircraft that pilot flies
LOCATION	-	pilot's base (integer code)
SALARY	-	pilot's salary

The following validations are required:

SURNAME	-	must be alphabetic, but may contain a space, hyphen, or apostrophe
DOBIRTH	-	pilot must be over 21 years of age
LOCATION	-	must be one of code values: 1, 7, 11, 12, or 13
SALARY	-	must be between 25,000 and 100,000

This system also needs a feature that allows you to enter abbreviations in the *AIRCRAFT* field. For example, instead of entering `BOEING 747`, you could enter `747`, and `BOEING 747` would be the actual field value.

Sample Function Definitions

This section contains sample functions for *PREVAL*, *POSTVAL*, *VALIDATE*, and *TINKER*.

```
     ∇ PREVAL
[1]    ABBREV← 3 3 ρ'TRDD10747'
[2]    FULLNAME←3 22 ρ(22↑'HAWKER SIDDELEY TRIDENT'),(22↑'MCDONNELL
       DOUGLAS DC10'),22↑'BOEING 747'
[3]    VALFLDS←CONVFLDS 'SURNAME,BIRTHDATE,AIRCRAFT,SALARY,LOCATION'
[4]    TODAY←100⊥100|3↑⎕TS
     ∇

     ∇ POSTVAL;T
[1]    T←6 ⎕FD 'ABBREV FULLNAME VALFLDS TODAY'
     ∇

     ∇ R←VALIDATE A
[1]    R←1
[2]    →(VALFLDS=A)/L1,L2,0,L3,L4
[3]    →0
[4]    L1:R←∧/INP∊' ABCDEFGHIJKLMNOPQRSTUVWXYZ''-' ◇ →0
[5]    L2:R←21≤⌊0.0001×TODAY-INP ◇ →0
[6]    L3:R←(INP≥25000)∧INP<100000 ◇ →0
[7]    L4:R←INP∊ 1 7 11 12 13 ◇ →0
     ∇

     ∇ R←TINKER A;T
[1]    R←1
[2]    →(A≠VALFLDS[3])/0
[3]    T←ABBREV∧.=3↑INP
[4]    →(1∊T)↓0
[5]    INP←,T⌿FULLNAME
     ∇
```

APL CONSTRAINTS

Often in MABRA systems you cannot identify records by standard MABRA constraints, but must apply APL expressions to field values. For example, in a system that contains the fields, *SALES* and *EXPENSES*, you might want to find all records that represent a **profit** (0<*SALES-EXPENSES*). You cannot use a standard MABRA constraint unless the field *PROFIT* exists, and undergoes evaluation in workspace 85 *MABRAUTIL*.

MABRA, however, allows you to enter an APL expression at the *CONSTRAINTS* prompt. The following expression,

```
      ⍎0<∆SALES-∆EXPENSES
```

operates on *SALES* and *EXPENSES* to identify the records that show a profit. Note that the expression must be prefixed by a hydrant (⍎); all fields must be prefixed by a delta (∆); all fields must be spelled in full; the result must be binary. The APL variables (in this case, ∆*SALES* and ∆*EXPENSES*) always form an array of rank 2.

INDEX

A

- abbreviated data in field (also see field code status); 2 11
- absolute record numbers; 55 56
- *AC*; 35 37 38
  - ex.; 82 83
- *ACC*; access file; 103 107
- access; 2 4 10 40 42 46-54 56 85-87 89 103 107-109
- active users of 86 *MABRA*; per *ST*; 46
- adding codes to field's code-desc. table; *AC*; 35 37 38
- adding field orders with *FO*; *A*; 44
- adding fields; *AF*; 8
- adding records; *AR*; 14
- *AF*; 8 11 48 101
  - ex.; 59 65 101
- *ALL*; 8 20 21 25 49 50-52 93 95
- α; constraint with "alpha" prefix; 18 21
- *AN*; 16 31-34 47 49 109
  - ex.; 69 72-79
- ∧; constraint with "and" connective; 18 19
- APL's immediate-execution mode; 1 2 4
- APL constraints; 118
- *APPEND* fn.; 85 *MABRAUTIL*; 90 95
- *AR*; 14 27 29 35 40 43 45 47 49 57 108 110 111 115
  - ex.; 2 14 60 64 89
- ←; "assignment" APL symbol; 22
- attributes of a record (see field)
- averaging; kind of *AN*; 32

B

- batch printing of records; *BP*; 25 109
- binary data type; 1 8 9 14
- blank code description; with *AC*; 35
- block size; 40 104
- blocking factor; of master file; 40 41 45 56 104
- blot; for *NEWPASS*; 54
- boolean data (see binary data type)
- *BOOLREL* var.; related fields; 110 111 113
- *BP*; 14 16 25-27 35 47 49 55 106 109
- [ ]; constraint with "brackets" or "subscripts"; 19
- →; "branch" APL symbol; 15
- B-task; for *BP*; 26 27
  - for *UP*; 27-29 60 103 105 110

C

- *CA*; 48 51-54
- *CCURSYSNUM* var.; current system number (85 *MABRAUTIL*); 92
- *CD*; 14 16 23 24 27 29 35 47 49 57 95 108 110 111 115
  - ex.; 66
- *CF*; 8 11 12 35-37 48 56 101
  - ex.; 67 68 80 81 102
- *CHANGETYPE* fn.; field's data type (85 *MABRAUTIL*); 102
- changing a date field's format; *DDMMYY* fn. in 85 *MABRAUTIL*; 90 100
- changing a field's data type; 85 *MABRAUTIL*; 101 102
- changing access; *CA*; 48 51
- changing field definitions; *CF*; 11 35
- changing field orders with *FO*; *C*; 44 45
- changing records having diff. values in a field; *CD*; 23 24
- changing records having the same values in a field; *CR*; 21 22
- character data type; 1 8 9 14
- character string; 16
- *CMDACC* var.; *ACC* file; 107
- *CODE* var.; *DIR* file; 106
- coded field (also see field code status); 2 11 35 36 80 81 104 106
- code-description table (also see coded field); 35-39 45 72 98 104
- code-description table; ex.; 82 83
- codes' descriptions; *GETDESC* fn. in 85 *MABRAUTIL*; 90
- :; "colon" symbol (see command prompt); 4
- column; 1
- command access; *ACC* file; 107
  - with multi-user MABRA; 47 48 51 52 85 86
- command prompt in Data Mode; nn:; 7
- command prompt in System Mode; :; 4 58
- ~; "complement" or "negation" of constraint; 20
- complement (~); complement or negate a field name; 21
- *CONS* var.; *UPD* file; 108
- constraints (see 86 *MABRA* constraints and APL constraints)
- ⊃; constraint that "contains" a value; 16 19
- copying a MABRA system; *CS*; 40 42
- *CR*; 14 16 21-23 27 29 35 47 49 55 57 108 110 111 115
  - ex.; 62

- *CR* versus *CD*; how to save money; 57
- *CREATE*; 4 5 8
  - ex.; 58
- *CREATOR*; 2 4 5 10 41 47 48 50 51 54 91 103 105 109
  - ex.; 87 90 92
- critical commands; *AF*, *CF*, *DF*, *RS*, *RB*, *CS*; 48 105
- crosstabulation analysis; *AN*; 2 31-34 69 109
- crosstabulation parameters; per *AN*; 32
- *CS*; 40 42 43 47 48
- *CURSYSNAME* var.; current system name (85 *MABRAUTIL*); 92

D

- Data Mode; 86 *MABRA*; 3 7 8 109
- date data type; 1 8 9 14 18
- date field's format (also see changing a date field's format); 9 14 90 100
- *DATEFLAG* var; 9
- *DC*; 35 38 39
- *DDMMYY* fn.; 85 *MABRAUTIL*; 90 100
- debugging hints; related fields; 114
- *DELETE* fn.; 85 *MABRAUTIL*; 90 97
- deleted records; 24 41 45 56 104 105
- deleting codes from code-desc. table; *DC*; 35 38 39
- deleting field orders with *FO*; *D*; 44 45
- deleting fields; *DF*; 11
- deleting records; *DR*; 24
- Δ; "delta" prefix for 85 *MABRAUTIL*'s global variables; 94-96 99 118
  - "delta" print field for absolute record numbers; 55
- *ΔFLDNAMES* var.; names of vars. in master file (85 *MABRAUTIL*); 92
- ⍙; "delta-underscore" prefix for variable of field names; 106
- *DESC* var.; *DIR* file; 106
- *DF*; 8 11 48 101
  - ex.; 68 89 102
- ◇; "diamond" feature for stacking input; 2 56 57 63
- *DIR*; field input validation per comp. 6; 115
  - miscellaneous directory file; 35 91 103-106 108
  - related fields per comp. 17; 110
- *DIRTY* var.; directory file's tie number (85 *MABRAUTIL*); 92
- *DR*; 14 16 24 25 27 29 35 41 47 49 55 108
  - ex.; 63

E

- 85 *MABRAUTIL*; 2 11 90 91 93 101 118
  - how to use; 92 102
- 86 *MABRA*; 4 101 109
  - constraints; 16-20 23 32 61 93
  - Data Mode; 3
  - how to use; 58 69 80 85 87 90 102
  - on-line assistance; 2
  - status of system with *ST*; 45 46
  - System Mode; 3
- elements in a field; APL ρ; 9
- *ENTER*; 86 *MABRA*; 4 7 8 109
  - 86 *MABRA* ex.; 58 69 80 85 87 90 101 102
- *ENTER* fn.; 85 *MABRAUTIL*; 90 91 92 98
- entering data in a field; with *AR* or *CD*; 14 23 24
- =; constraint "equal to" a value; 16
- *ERASE*; 4 6
- event numbers (see *⎕TRAP* error conditions)
- examples showing how to use 86 *MABRA*; 58-89
- *EXIT* fn.; 85 *MABRAUTIL*; 90 98
- exiting 86 *MABRA*; 69 79 84 86 89 91
  - abnormally; 105

F

- field; 1
- field access; *ACC* file; 107
  - for *AR*, *CR*, *CD*, *LR* with multi-user MABRA; 10 47-49 51-53 85 86
- field access to add values with *AR*; for multi-user MABRA; 47 49 53
- field access to change values with *CR* and *CD*; for multi-user MABRA; 47 49 53
- field access to list values with *LR*; for multi-user MABRA; 47 50 53
- field averaging with *AN*; *A*; 33 34
- field code status (also see coded field); 2 8 10-12 13 35 36
- field data type; changing with *CF*; 35 101 102
  - character, integer, real, binary, date; 1 8-11 14 40 104
  - input validation; 115
- field definitions; attributes of a field; 8 10 101
- field formats; for *LR*, *CR*, *DR*, and *BP* output; 21
- field input validation; 105 115-117
- field inversion status (also see inverted field); 8 10-13 35 36
- field management commands; Data Mode; 3 8
- field name; 8 10-12 21 35 36
  - abbreviate; 10 61
- field name derivatives; for *AN*; 32
  - for *LR*, *CR*, *DR*, and *BP*; 20 21 25 26
- field ordering for *AR*; *FO*; 43-45 57 106
- field orders' identification numbers with *FO*; 44 45
- field output format; *⎕FMT*; 8 9 10 11 12 13 35 36 104
- field totalling and averaging with *AN*; *TA*; 33
- field totalling with *AN*; *T*; 33 34
- field type (see field data type)
- field width; 8-13 35 36

FILE ACCESS ERROR; 51
file documentation; 103-109
file status; per *ST*; 45
filler data; for empty field; 10 14 15
FLDACC var.; *ACC* file; 107
FLDDESCS var.; coded field's descriptions; 92 104 106
FLDFMTS var.; fields' ⎕*FMT* arguments; 92 104
FLDNAMES var.; field names; 92 93 98 99 104 106-108 110 111 115 116
FNTY var.; tie number of MABRA function file (85 *MABRAUTIL*); 92
FO; 40 43-45 57
FOUNDMASK var.; records in constraint; 93 94 96 97 108
freq. count., field tot., field avg. with *AN*; *FTA*; 33
freq. count. and field avg. with *AN*; *FA*; 33
freq. count. and field tot. with *AN*; *FT*; 33
frequency counting; kind of *AN*; 32
frequency counting with *AN*; *F*; 33
function file (see MABRA function file)

G

GETDESC fn.; 85 *MABRAUTIL*; 90 98
global variables; 85 *MABRAUTIL*; 91 92 93 94 95 96 98 99
GRADE fn.; 85 *MABRAUTIL*; 90 99
>; constraint "greater than" a value; 16
≥; constraint "greater than or equal to" a value; 16

H

HELP; 2 8
  ex.; 87
how to save money; 55-57
how to use 86 *MABRA* (see 86 *MABRA*)
HSPRINT program (also see *BP*); 26 27
⍎; "hydrant" APL symbol for execute; 118

I

IND var.; *UPD* file; 108
initiating an update task; *UP*; 27
INP var.; field input validation; 115 116
input validation (see field input validation)
integer data type; 1 8 9 14
INV; inverted fields file; 91 103 104 107
inverted field (also see field inversion status); 11 45 56 104 107
INVTY var.; inverted file's tie number (85 *MABRAUTIL*); 92

J

"jump to" feature; see → ("branch" APL symbol); 15 43 63

K

kinds of crosstab. analyses with *AN*; 32

L

LA; 48 50
  ex.; 86
LC; 35 38 47
  ex.; 72 82-84 90 91
<; constraint "less than" a value; 16
≤; constraint "less than or equal to" a value; 16
LF; 8 10 43 44 47
  ex.; 59 65 68 69 80 81 85 87 88 90
LIST; 4 5
  ex.; 58 87
listing a code-description table; *LC*; 35 38
listing access; *LA*; 48 50
listing an update request; *LU*; 29 30
listing field orders with *FO*; *L*; 44 45
listing fields; *LF*; 10 11
listing records; *LR*; 20
LR; 14 16 20 35 47 49 50 55 66 99
  ex.; 61-63 65 67 68 70 71 80 81 83-85 88 89 91
LU; 14 29 30 47

M

MABRA; computerized program (levels of control); 1
  definition; 1
  file documentation ; 103-109
  function file; 91
  structure of record; 1
  version number; 105
  workspaces (see 85 *MABRAUTIL* and 86 *MABRA*
  worthwhile features; 1 2
MAS; master file; 91 103-107 110 111
MASBLKSIZE var.; numb. records per master comp.; 92 104 106 107
MASFSZ var.; number of components in the master file (85 *MABRAUTIL*); 92
MASTY var.; master file's tie number (85 *MABRAUTIL*); 92
MAX var.; *DIR* file; 106
maximum number of records expected in system; per *ST*; 45

MAXNUMRCDS var.; *DIR* file; 105
miscellaneous commands; *RB*, *RS*, *CS*, *FO*, *ST*; 40
MSK; mask file; 91 103 108 109
MSKTY var.; mask file's tie number (85 *MABRAUTIL*); 92
multi-user MABRA; 47 48 85-89

N

negate (~); negate or complement a field name; 21
~; "negation" or "complement" of constraint; 20
new sign-on password (also see *NEWPASS*); 54
new snapshot; *NS*; 28 29
NEWPASS; 48 54 103
non-interactive workspace (see 85 *MABRAUTIL*)
≠; constraint "not equal to" a value; 16
NS; 14 27-29 47 109
    ex.; 60 62 63 66
N-task; for *AN*; 109
    for *BP*; 26 27
    for *UP*; 27-29 60 63 64 66 103 105 110 114
number of elements in field; APL ρ; 9
number of pending update requests; per *ST*; 45
NUMRCDS var.; number of records in a system; 92 93 97 104 106 108 109

O

∨; constraint with "or" connective; 18 19
output format; ⎕*FMT* (see field output format)

P

( ); constraint with "parentheses"; 18 19
    "parentheses" (cause code desc. output for *LR*, *CR*, *DR*, and/or *BP*); 21 35
password for signing on the APL system (also see *NEWPASS*); 54
POSTUPD fn.; related fields; 110-114
POSTVAL fn.; related fields; 114-117
PREUPD fn.; related fields; 110-114
PREVAL fn.; related fields; 114-117
PREVIOUS constraint; 8 20 23 56 109
print fields; for *LR*, *CR*, *DR*, and *BP*; 20 21 24-26
printing records; ex.; 69-71 80 81
purging deleted records; *RB*; 41 56
purging processed update requests (see *UPD*)

Q

⎕*BOUNCE*; 114
⎕*FMT* (see field output format)
⎕*PW*; for *LR*; 21
⎕*REPLACE*; *MAS* file; 107
⎕*RUNS*; related fields; 114
⎕*SP*; related fields; 114
⎕*TRAP* error conditions; event numbers; 92 94 95 96 99 100
⎕*TS*; related fields; 114
⎕*WA*; workspace available; 93
QUIT; 2 4 8 91
quotations; 16
⍞; "quote-quad" APL symbol for data input; 116

R

RA; 48 50 51
range of update requests for *WU*; 29 30
range points for *AN*; 31 32 33
RB; 24 40 41 48 56 97 105
RCDACC var.; *ACC* file; 107
READ fn.; 85 *MABRAUTIL*; 90 93-96 98 99
READPTR var.; 85 *MABRAUTIL*; 93 94
real data type; 1 8 9 14 34
reblocking the master file; *RB*; 40 41 56
record access; *ACC* file; 107
    with multi-user MABRA; 47-49 51 52 85 86
record management commands; Data Mode; 3 8 14
record of data; 1
record size; 56
    how to derive; 40
    per *ST*; 45
related fields; 106 110-114
removing access; *RA*; 48 50 51
RENAME; 4 6
REORD fn.; 85 *MABRAUTIL*; 90 99 100
REPLACE fn.; 85 *MABRAUTIL*; 90 93 95-97
REPLACEPTR var.; 85 *MABRAUTIL*; 93 96
report formatting facility; ⎕*FMT* (see field output format)
resizing the master file; *RS*; 41
rounding real numbers; per *AN*; 34
row; 1
RS; 5 40 41 42 48 105

S

SA; 48-51 54
  ex.; 85 86
search key; 2 11 21 56 107
SEARCHFOR fn.; 85 *MABRAUTIL*; 90 93 94 96-98
setting access; *SA*; 48-50
shared access (also see multi-user MABRA); 2 4
sort fields; for *LR*, *CR*, *DR*, and/or *BP*; 21 25 26 64 99
sort key; as leftmost field name (also see search key); 21
sorting records; *GRADE* fn. in 85 *MABRAUTIL*; 90 99
  86 *MABRA* ex.; 69-71 80-82
*SORTIN*xx; file for *BP*; 27
*SORTOUT*xx; file for *BP*; 27
SORTREQ program (also see *BP*); 27
source field; changing a field's data type; 101
source system; per *CS*; 42 43
ST; 40 45 46
stacking input; see: ◇ ("diamond" feature); 2 56 57 63
START; 4 9
  ex.; 58 69 80 85 87 90 101 102
status of MABRA system; *ST*; 45 46
STOP; 4 7 8 15
  ex.; 69 84 89
structure of a record; 1
SUBMITREQUEST fn.; *SYS* file; 103
[ ]; constraint with "subscripts" or "brackets"; 19
suppress certain field prompts; per *FO*; 43
SYS; system directory file; 103
SYSACC var.; *SYS* file; 103
SYSNAMES var.; *SYS* file; 103
SYSNUMS var.; *SYS* file; 103
System Mode; 86 *MABRA*; 3 4 48 58
SYSUSERS var.; *SYS* file; 103

T

taking a new snapshot; *NS*; 28
target field; changing a field's data type; 101
target system; per *CS*; 42 43
temporary field name; changing a field's data type; 101
TENPERC fn.; 85 *MABRAUTIL*; 97
TINKER fn.; field input validation; 115-117
total number of fields; 1 45
total number of records; 1 41 45 105
total number of update requests; per *ST*; 46
totalling; kind of *AN*; 32
T-task; 26 28 109 114
TYPE var.; *UPD* file; 108
types of data (see field data type)

U

un-coded field (also see field code status); 35-37
UP; 14 15 27 29 30 47 48 54 56 109
  ex.; 60 62 63 64 66 89
UPD; update request file; 28 103 105 108
update request; *AR*, *CR*, *CD*, *DR*, *WU* requests in *UPD* file; 108
  organizing; 57
update request (also see *UP*); 15 22 24 25 27 28 45 46 60 62-64 66 89 106
update task; also see *UP*; 27-29 48 109
  parameters of *DIR* file; 105 106
  structure (related fields); 110
updating records; *REPLACE* fn. in 85 *MABRAUTIL*; 90
utility workspace (see 85 *MABRAUTIL*)

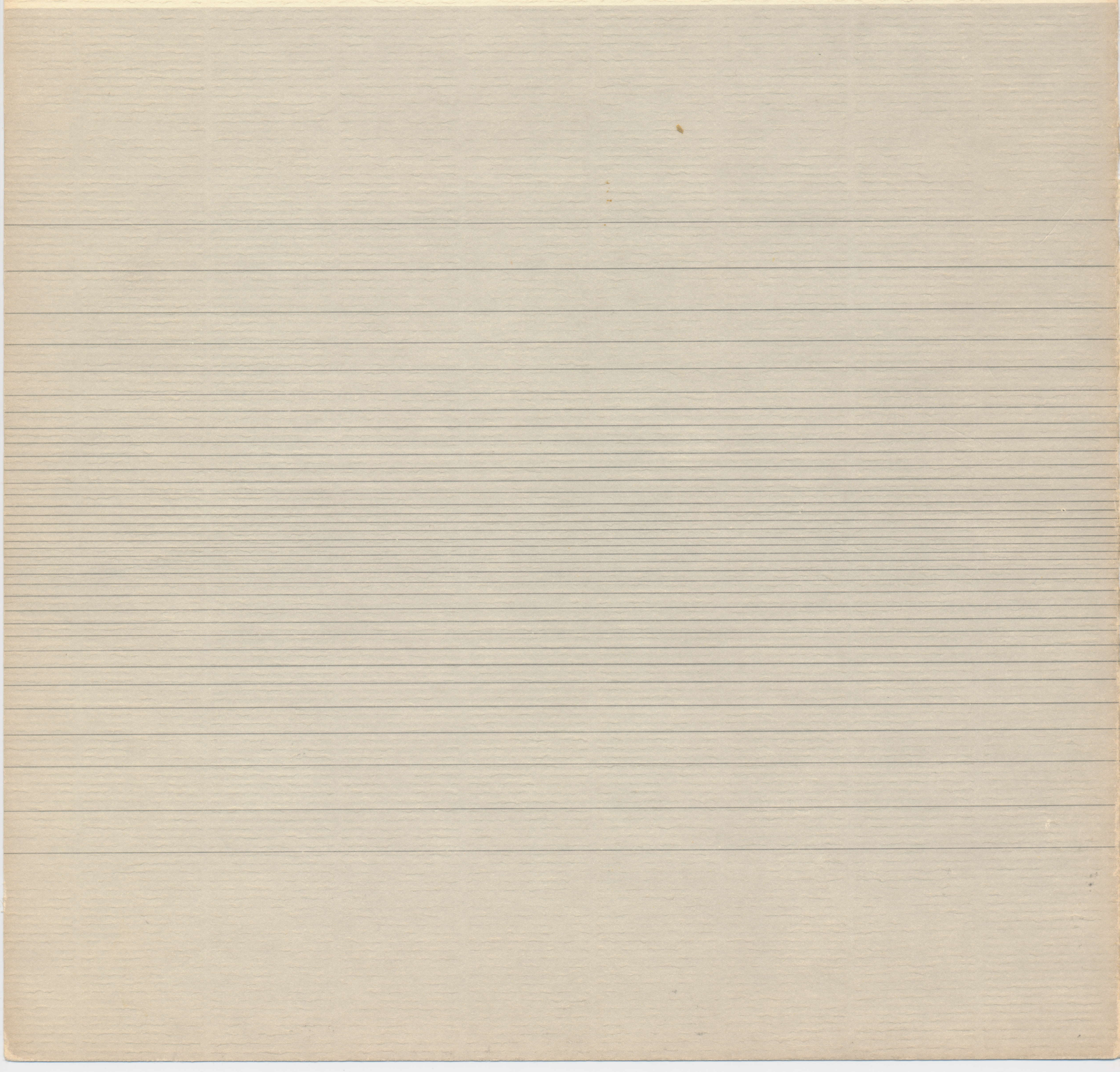
V

VALIDATE fn.; field input validation; 115-117
validation (see field input validation)
validation indicators; field input validation; 115 116
various field orders; per *FO*; 43
version number; MABRA's; 105

W

WHENUPD fn.; related fields; 114
WHOUPD fn.; related fields; 114
width (see field width)
width of codes' descriptions; 36 37 104
withdrawing an update request; *WU*; 29
workspace (also see 85 *MABRAUTIL* and 86 *MABRA*); 23 40 90 93 96 114
WS FULL error; 93 96
WU; 14 29 30 47 108

Y

YYMMDD; kind of format for date field; 100